
# ADAPTIVE 2010

The Second International Conference on Adaptive and Self-Adaptive Systems and Applications

November 21-26, 2010 - Lisbon, Portugal

**ComputationWorld 2010 Editors**

Ali Beklen, IBM Turkey, Turkey
Jorge Ejarque, Barcelona Supercomputing Center, Spain
Wolfgang Gentzsch, EU Project DEISA, Board of Directors of OGF, Germany
Teemu Kanstren, VTT, Finland
Arne Koschel, Fachhochschule Hannover, Germany
Yong Woo Lee, University of Seoul, Korea
Li Li, Avaya Labs Research - Basking Ridge, USA
Michal Zemlicka, Charles University - Prague, Czech Republic

# ADAPTIVE 2010

## Foreword

The Second International Conference on Adaptive and Self-adaptive Systems and Applications [ADAPTIVE 2010], held between November 21 and 26 in Lisbon, Portugal, targeted advanced system and application design paradigms driven by adaptiveness and self-adaptiveness. With the current tendencies in developing and deploying complex systems, and under the continuous changes of system and application requirements, adaptation is a key feature. Speed and scalability of changes require self-adaptation for special cases. How to build systems to be easily adaptive and self-adaptive, what constraints and what mechanisms must be used, and how to evaluate a stable state in such systems are challenging duties. Context-aware and user-aware are major situations where environment and user feedback is considered for further adaptation.

We take here the opportunity to warmly thank all the members of the ADAPTIVE 2010 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ADAPTIVE 2010. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ADAPTIVE 2010 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ADAPTIVE 2010 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the areas of adaptive and self-adaptive systems and applications.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the beautiful surroundings of Lisbon, Portugal.

ADAPTIVE 2010 Chairs:

Radu Calinescu, Aston University, UK
Marc Gilg, University of Haute Alsace, France
Leszek Holenderski, Philips Research - Eindhoven, The Netherlands
Weirong Jiang, Juniper Networks, USA
Stefan Mangold, Disney Research - Zurich, Switzerland
Thomas H. Morris, Mississippi State University, USA
Naoki Wakamiya, Osaka University, Japan

# ADAPTIVE 2010

## Committee

**ADAPTIVE Advisory Chairs**

**Academia**
Radu Calinescu, Aston University, UK
Marc Gilg, University of Haute Alsace, France
Thomas H. Morris, Mississippi State University, USA
Naoki Wakamiya, Osaka University, Japan

**Industry**
Weirong Jiang, Juniper Networks, USA
Leszek Holenderski, Philips Research - Eindhoven, The Netherlands
Stefan Mangold, Disney Research - Zurich, Switzerland

**ADAPTIVE 2010 Technical Program Committee**

Habtamu Abie, Norwegian Computing Center - Oslo, Norway
Muhammad Tanvir Afzal, Technical University Graz, Austria
Mahmood Ahmadi, TU Delpt, The Netherlands
Giner Alor Hernández, Instituto Tecnológico de Orizaba - Veracruz, México
John Atkinson, Universidad de Concepcion, Chile
Marco Benini, Università degli Studi dell'Insubria, Italy
Claudio Biancalana, LAit S.p.A., Italy
Dieter Blomme, Siruna NV / Ghent University, Belgium
Rajendra V. Boppana, University of Texas at San Antonio, USA
Radu Calinescu, Aston University, UK
Aldo Campi, DEIS / University of Bologna - Cesena, Italy
Bogdan Alexandru Caprarescu, West University of Timisoara, Romania
Eduardo Cerqueira, Federal University of Para, Brazil
Kuan-Ta Chen, Academia Sinica, Taiwan
Yan Chen, Georgia State University, USA
Po-Hsun Cheng, National Kaohsiung Normal University, Taiwan
Jose Alfredo F. Costa, Federal University (UFRN), Brazil
Carlos Cuesta, Rey Juan Carlos University, Madrid, Spain
Vieri del Bianco, University College Dublin, Ireland
Michel Diaz, LAAS - Toulouse, France
Petre Dini, Concordia University, Canada / IARIA, USA
Mihaela Dinsoreanu, Technical University of Cluj-Napoca, Romania
Ioanna Dioysiou, University of Nicosia, Cyprus
Christian Doerr, TU Delft, The Netherlands
Pierpaolo Dondio, Trinity College Dublin, Ireland
Schahram Dustdar , Vienna University of Technology, Austria

Larbi Esmahi, Athabasca University, Canada
Ernesto Exposito, INSA/DGEI and LAAS/CNRS, France
Alois Ferscha, Johannes Kepler Universität Linz, Austria
Ziny Flikop, Consultant, USA
Carlos Flores, Universidad de Colima, México
Mohammad Muztaba Fuad, Winston-Salem State University, USA
Francisco José García Peñalvo, Universidad de Salamanca, España
Harald Gjermundrod, University of Nicosia, Cyprus
George Giannakopoulos, University of Trento, Italy
Georgios K. Giannikis, University of Thessaly - Volos, Greece
Marc Gilg, University of Haute Alsace, France
Simone Grassi, Trinity College Dublin, Ireland
Jon Hall, Open University, UK
Leszek Holenderski, Philips Research - Eindhoven, The Netherlands
Vera Hollink, Centrum voor Wiskunde en Informatica - Amsterdam, The Netherlands
Jianmin Jiang, University of Bradford, UK
Weirong Jiang, Lawrence Livermore National Laboratories, USA
Iliya S. Kabak, Stankin Moscow State Technological University, Russia
Elsy Kaddoum, IRIT - Université Paul Sabatier, France
Hamid Reza Karimi, University of Agder - Grimstad, Norway
John Keeney, Trinity College Dublin. Ireland
Serge Kernbach, University of Stuttgart, Germany
Rico Kubster, University of Kassel, Germany
Kiran Lakkaraju, Sandia National Laboratories, USA
Mikel Larrea, The University of the Basque Country, Spain
Jingpeng Li, The University of Nottingham, UK
Robert Logie, Osaka Gakuin University, Japan
Luca Longo, Trinity College Dublin, IrelandEmiliano Lorini, Institut de Recherche en Informatique de Toulouse (IRIT), France
Josef Makolm, Federal Ministry of Finance - Vienna, Austria
Stefan Mangold, Disney Research - Zurich, Switzerland
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal
Kevin McCarthy, University College Dublin (UCD), Ireland
Mehdi Mekni, Laval University / Fujitsu, Canada
Tommy Morris, Mississippi State University, USA
Gero Mühl, University of Rostock, Germany
Roel Ocampo, University of the Philippines Diliman, Philippines
Sverrir Olafsson, Reykjavik University, Iceland
Flavio Oquendo, Université Européenne de Bretagne - UEB | Université de Bretagne-Sud - VALORIA, France
Dalimir Orfánus, ABB Corporate Research Center - Oslo, Norway
Béatrice Paillassa, Universite de Toulouse, France
Athanasios Panagopoulos, National Technical University of Athens, Greece
Susan Pancho-Festin, University of the Philippines, Philippines
Stefan Poslad, Queen Mary University of London, UK
Claudia Raibulet, University of Milano-Bicocca, Italy
Sebastian Ries, TU Darmstadt, Germany
Sita Ramakrishnan, Monash University, Australia

**Copyright Information**

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

**Table of Contents**

# *SERSCIS-Ont*

A Formal Metrics Model for Adaptive Service Oriented Frameworks.

Mike Surridge, Ajay Chakravarthy, Maxim Bashevoy, Martin Hall-May
IT Innovation Centre, 2 Venture Road, Chilworth, Southampton, UK.
{ms,ajc,mvb,mhm}@it-innovation.soton.ac.uk

*Abstract*— **In the Future Internet, programs will run on a dynamically changing collection of services, entailing the consumption of a more complex set of resources including financial resources. The von Neumann model offers no useful abstractions for such resources, even with refinements to address parallel and distributed computing devices. In this paper we detail the specification for a post-von Neumann model of metrics where program performance and resource consumption can be quantified and encoding of the behaviour of processes that use these resources is possible. Our approach takes a balanced view between service provider and service consumer requirements, supporting service management and protection as well as non-functional specifications for service discovery and composition.**

***Keywords-adaptive metrics; SOA; measurements;constraints; QoS***

## I. INTRODUCTION

A (relatively) open software industry developed for non-distributed computers largely because of the von Neumann model [8], which provided the first practical uniform abstraction for devices that store and process information. Given such an abstraction, one can then devise models for describing computational processes via programming languages and for executing them on abstract resources while controlling trade-offs between performance and resource consumption. These key concepts, resource abstraction supporting rigorous yet portable process descriptions, are fundamental to the development and widespread adoption of software assets including compilers, operating systems and application programs.

In the Future Internet, programs will run on a dynamically changing collection of services, entailing the consumption of a more complex set of resources including financial resources (e.g. when services have to be paid for). The von Neumann model offers no useful abstractions for such resources, even with refinements to address parallel and distributed computing devices. In this context, we need something like a 'post-von Neumann' model of the Future Internet of Services (including Grids, Clouds and other SOA), in which: program performance and consumption of resource (of all types) can be quantified, measured and managed; and programmers can encode the behaviour of processes that use these resources, including trade-offs between performance and resource consumption, in a way that is flexible and portable to a wide range of relevant resources and services.

In this paper, we describe the metric model developed within the context of the SERSCIS project. SERSCIS aims to develop adaptive service-oriented technologies for creating, monitoring and managing secure, resilient and highly available information systems underpinning critical infrastructures. The ambition is to develop technologies for such information systems to enable them to survive faults, mismanagement and cyber-attack, and automatically adapt to dynamically changing requirements arising from the direct impact of natural events, accidents and malicious attacks. The proof of concept (P-o-C) chosen to demonstrate the SERSCIS technologies is an airport-based collaboration and decision-making scenario. In this scenario, separate decision makers must collaborate using a number of dynamic interdependent services to deal with events such as aircraft arrival and turn-around, which includes passenger boarding, baggage loading and refuelling. The problem that decision makers face is that the operations are highly optimised, such that little slack remains in the turnaround process. If a disruptive event occurs, such as the late arrival of a passenger, then this has serious knock-on effects for the rest of the system that are typically difficult to handle.

The focus for our work is therefore to support the needs of both service providers and consumers. Our goal is to allow providers to manage and protect their services from misbehaving consumers, as well as allowing consumers to specify non-functional requirements for run-time service discovery and composition should their normal provider become unreliable. In this sense, SERSCIS-Ont combines previous approaches from the Semantic Web community focusing on service composition, and from the service engineering community focusing on quantifying and managing service performance.

The rest of the paper is organised as follows. Section II defines and clarifies the terminology used for metrics, measurements and constraints. In Section III we present the SERSCIS-Ont metric model. Here each metric is discussed in a detail along with the constraints which can be imposed upon these metrics. Section IV reviews the state of the art for related work and compares and contrasts research work done in adaptive system metrics with SERSCIS-Ont. Section V presents the results of the validation/simulation experiment carried out to test the applicability of the SERSCIS metrics. Finally we conclude the paper in Section VI

## II. METRICS MEASUREMENTS AND CONSTRAINTS

It is important to distinguish between the terminology used for metrics, measurements and constraints. In Figure **1** we show the conceptual relationships between these terms.

 ISBN: 978-1-61208-109-0 

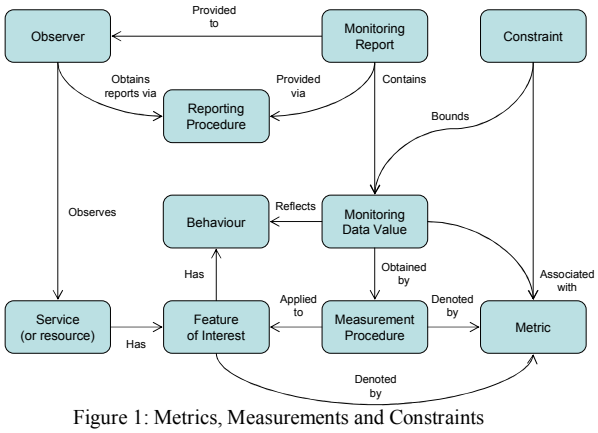



Figure 1: Metrics, Measurements and Constraints

*Services* (or sometimes the *resources* used to operate them) are monitored to provide information about some *feature of interest* associated with their operation. The *monitoring data* by some *measurement procedure* applied to the feature of interest at some time or during some time period. *Metrics* are labels associated with this data, denoting what feature of interest they refer to and (if appropriate) by which measurement procedure they were obtained. Finally, monitoring data is supplied to *observers* of the service at some time after it was measured via *monitoring reports*, which are generated and communicated to observers using a *reporting procedure*. It is important to distinguish between monitoring data for a feature of interest, and its actual *behaviour*. In many situations, monitoring data provides only an approximation to the actual behaviour, either because the measurement procedure has limited accuracy or precision, or was only applied for specific times or time periods and so does not capture real-time changes in the feature of interest. *Constraints* define bounds on the values that monitoring data should take, and also refer to metrics so it is clear to which data they pertain. Constraints are used in *management policies*, which define management actions to be taken by the service provider if the constraints are violated. They are also used in *SLA terms*, which define commitments between service providers and customers, and may specify actions to be taken if the constraints are violated. Note that management policies are not normally revealed outside the service provider, while SLA terms are communicated and agreed between the service provider and customer. Constraints refer to the behaviour of services or resources, but of course they can only be tested by applying some *testing procedure* to the relevant monitoring data. The testing procedure will involve some mathematical manipulation to extract relevant aspects of the behaviour from the monitoring data.

## III. SERSCIS METRICS

In SERSCIS, we aim to support metrics which will represent the base classes that capture the physical and mathematical nature of certain kinds of service behaviors and measurements. These are described below.

### A. Absolute Time

This metric signifies when (what time and date) some event occurs. It can be measured simply by checking the time when the event is observed. Subclasses of this metric would be used to refer to particular events, e.g. the time at which a service is made available, the time it is withdrawn from service, etc. There are two types of constraints imposed on this metric. (1) a lower limit on the absolute time, encoding “not before” condition on the event. (2) an upper limit on the absolute, encoding a “deadline” by which an event should occur.

### B. Elapsed Time

This metric just signifies how long it takes for some event to occur in response to some stimulus. It can be measured by recording the time when the stimulus arises, then checking the time when the subsequent event is observed and finding the difference. Subclasses of this metric would be used to refer to particular responses, e.g. the time taken to process and respond to each type of request supported by each type of service, or the time taken for some internal resourcing action such as the time for cleaners to reach an aircraft after it was scheduled and available. In the SERSCIS P-o-C, it should be possible to ask a consumer task for the elapsed times of all responses corresponding to the metric, and possibly to ask for the same thing in a wider context (e.g. from a service or service container). Constraints placed on elapsed time are (1) an upper limit on the elapsed time which encodes a lower limit on the performance of a service. (2) a lower limit which is typically used only in management policies to trigger actions to reduce the resource available if a service over-performs. If there are many events of the same type, one may wish to define a single constraint that applies to all the responses, so if any breaches the constraint the whole set is considered to do so. This allows one to test the constraint more efficiently by checking only the fastest and slowest response in the set. Sometimes it may be appropriate to define constraints that include more than one response time. For example, suppose a service supports aircraft refuelling but the amount of fuel supplied (and hence the time spent actually pumping fuel) is specified by the consumer – See Figure **2**.

 ISBN: 978-1-61208-109-0 

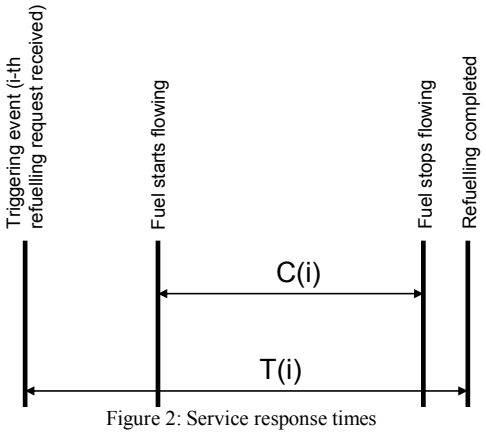


Figure 2: Service response times

In this situation the service provider can't guarantee the total response time T(i), because they have no control over the amount of time C(i) for which the fuel will actually flow into the aircraft. But they can control how long it takes for a fuel bowser to reach the aircraft after the refuelling request is received, and how long it takes to connect and disconnect the fuelling hoses and get clear after fuelling is completed, etc. So the service provider may prefer to specify a constraint on the difference between the two elapsed times. In SERSCIS, anything that is constrained should be a metric (to keep the SLA and policy constraint logic and schema simple), so in this situation one should define a new metric which might be called something like 'fuelling operation time'. One then has two options to *obtain its value (1) measure it directly so values are returned by the measurement procedure; or (2) define rules specifying the relationship between the new metric's value and the other metrics whose values are measured.*

### C. Counter

This metric signifies how often events occurs since the start of measurement. It can be measured by observing all such events and adding one to the counter (which should be initialised to zero) each time an event occurs. In some situations it may be desirable to reset the counter to zero periodically (e.g. at the start of each day), so the metric can refer to the number of events since the start of the current period. In this case it may be appropriate to record the counter for each period before resetting it the retained value for the next period. Subclasses of this metric would be used to refer to particular types of events, e.g. the number of requests of each type supported by the service, or the number of exceptions, etc. In the SERSCIS P-o-C, it should be possible to ask a consumer task, service or container for the counters for each type of request and for exceptions arising from each type of request. Note that some types of request may only be relevant at the service or container level, and for these the counters will only be available at the appropriate level. Constraints here are upper and lower limits encoding the commitments not to send too many requests or generate too many exceptions or to trigger management actions. There are also limits on the ration between the numbers of events of different types.

### D. Max and Min Elapsed Time

These metrics signify the slowest and fastest response to some stimulus in a set of responses of a given type, possibly in specified periods (e.g. per day). They can be measured by observing the elapsed times of all events and keeping track of the fastest and slowest responses in the set. Subclasses of this metric would be used to refer to particular types of response, e.g. times to process and respond to each type of service request, etc. In the SERSCIS P-o-C, it should be possible to ask a consumer task, service or container for the minimum and maximum elapsed times corresponding to the metric. Constraints on such metrics signify the range of elapsed times for a collection of responses. Only one type of constraint is commonly used: an upper limit on the maximum elapsed time, encoding a limit on the worst case performance of a service.

### E. Mean Elapsed Time

This metrics signifies the average response to some stimulus for responses of a given type, possibly in specified periods. It can be measured by observing the elapsed times for all such responses, and keeping track of the number of responses and the sum of their elapsed times: the mean is this sum divided by the number of responses. Subclasses of this metric would be used to refer to particular types of response, e.g. times to process and respond to each type of service request, etc. In the SERSCIS P-o-C, it should be possible to ask a consumer task, service or container for the mean elapsed time corresponding to the metric. Constraints on this metric are the same as those for the elapsed time metric.

### F. Elapsed Time Compliance

This metric captures the proportion of elapsed times for responses of a given type that don't exceed a specified time limit. Metrics of this type allow the distribution of elapsed times to be measured, by specifying one or more compliance metrics for different elapsed time limits (See Figure **3**).

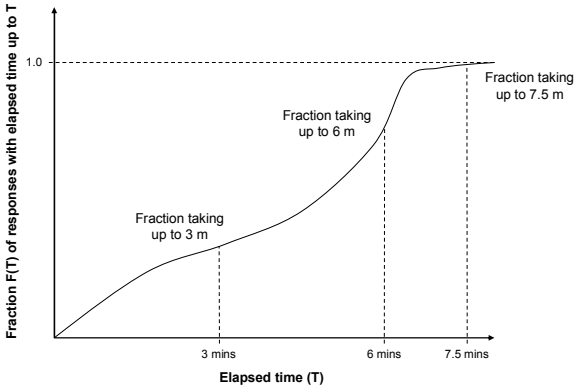


Figure 3: Elapsed time distribution

 ISBN: 978-1-61208-109-0 

When measuring elapsed time compliance, it is convenient to make measurements for all the metrics associated with a distribution like Figure 3. One has to observe the elapsed times for all relevant responses, and keep track of the number of responses that were within each elapsed time limit, and also the total number of responses. The value of the elapsed time compliance metric at each limit is then the ratio between the number of responses that didn't exceed that limit and the total number of responses. Subclasses of this metric would be used to refer to particular types of responses and time limits. For example, one might define multiple elapsed time compliance metrics for different time limits for responses to each type of request supported by the service, and for some internal process time. In the SERSCIS P-o-C, it should be possible to ask a consumer task, service or container for the elapsed time compliance for responses corresponding to the metric. It may also be useful to support requests for all elapsed time compliance metrics for a given type of response, allowing the compliance of the entire distribution function to be obtained at once. Note that some types of request may only be relevant at the service or container level, and for these the elapsed time distribution function will only be available at the appropriate level. Constraints for this metric are normally expressed as lower (and sometimes upper) bounds on the value of the metric for specific responses and time limits. SLA commitments typically involve the use of lower bounds (e.g. 90% of responses within 10 mins, 99% within 15 mins, etc), but both upper and lower bounds may appear in management policies (e.g. if less than 95% of aircraft are cleaned within 10 mins, call for an extra cleaning team).

*G. Non-recoverable resource usage and usage rate*

These metrics capture the notion that services consume resources, which once consumed cannot be got back again (this is what we mean by non-recoverable). In most cases, non-recoverable usage is linked to how long a resource was used, times the intensity (or rate) of usage over that period. It can be measured by observing when a resource is used, and measuring either the rate of usage or the total amount of usage at each observation. Subclasses of the non-recoverable usage metric would be used to refer to the usage of particular types of resources, for example on CPU usage, communication channel usage, data storage usage etc. In the SERSCIS P-o-C, it should be possible to ask a consumer task, service or container for the usage rate at the last observation, and the total usage up to that point. Ideally this should trigger a new observation whose result will be included in the response. The response should include the absolute time of the last observation so it is clear whether how out of date the values in the response may be. Non-recoverable resource usage is characterized by functions of the form:

$$U(S,t) \geq 0 \quad (1)$$

$$\frac{dU(S,t)}{dt} \geq 0 \quad (2)$$

$U$ represents the total usage of the non-recoverable resource by a set of activities $S$ up to time $t$. The range of $U$ is therefore all non-negative numbers, while the domain spans all possible sets of activities using the resource, over all times. In fact, $U$ is zero for all times before the start of the first activity in $S$ (whenever that may have been), and its time derivative is also zero for all times after the last activity has finished. The time derivative of $U$ represents the rate of usage of the non-recoverable resource. This must be well-defined and non-negative, implying that $U$ itself must be smooth (continuously differentiable) with respect to time, i.e. it can't have any instantaneous changes in value.

Constraints for non-recoverable usage and usage rate are typically simple bounds on their values. Both upper and lower bounds often appear in management policies to regulate actions to decrease as well as increase resources depending on the load on the service:

$$L_0 \leq U(S,t_0) - U(S,t_0) \leq L_1 \quad (3)$$

represents a constraint on the minimum and maximum total usage for a collection of activities $S$ in a time period from $t_0$ to $t_1$, while:

$$M_0 \leq \frac{dU(S,t)}{dt} \leq M_1, \forall t: t_0 \leq t \leq t_1 \quad (4)$$

represents a constraint on the maximum and minimum total usage rate for a collection of activities $S$ during a time period from $t_0$ to $t_1$. Note that it is possible to have a rate constraint (4) that allows a relatively high usage rate, in combination with a total usage constraint (3) that enforces a much lower average usage rate over some period. Alternatively, a contention ration could be introduced for usage rate constraints to handle cases where a resource is shared between multiple users but may support a high usage rate if used by only one at a time.

*H. Maximum and Minimum Usage Rate*

These metrics capture the range of variation in the usage rate (possibly in specified periods, which is described above. They can be measured by simply retaining the maximum and minimum values of the usage rate whenever it is observed by the measurement procedure. Subclasses of these metrics would be used to refer to maximum and minimum usage for particular types of resources. Constraints on maximum and minimum usage rate take the form of simple bounds on their values. Note that if we constrain maximum usage rate to be up to some limit, and the usage rate ever breaches that limit, then the constraint is violated however the usage rate changes later.

 ISBN: 978-1-61208-109-0 

*I. State*

This metric captures the current state of a service, with reference to a (usually finite) state model of the service's internal situation (e.g. the value of stored data, the status of supplier resources, etc). The value of the metric at any time must be a state within a well-defined state model of the service, usually represented as a string signifying that state and no other. It can be measured by observing the internal situation of the service and mapping this to the relevant state from the state model. In the SERSCIS PoC implementation, it should be possible to ask a task, service or container for its current state. Note that the state model of a service will normally be different from the state model of tasks provided by the service, and different from the state model of the container providing the service. State is an instantaneous metric – a measurement of state gives the state at the time of observation only. To obtain a measure of the history of state changes one should use state occupancy metrics or possibly non-recoverable usage metrics for each possible state of the service. Subclasses of the state metric will be needed to refer to particular state models and/or services. Constraints can be used to specify which state a service should be in, or (if the state model includes an ordering of states, e.g. security alert levels), what range of states are acceptable.

*J. State Occupancy*

This metric captures the amount of time spent by a task in a particular state (possibly in specified periods). It can be measured by observing state transitions and keeping track of the amount of time spent in each state between transitions. Note that for this to be practical one must predefine a state model for the task encompassing all its possible states, in which the first transition is to enter an initial state when the task is created.

The state of a resource on a service is a function of time:

$$S_i(t) \in \Sigma, \forall t \geq t_0 \qquad (5)$$

where $S_i(t)$ is the state of resource $i$ at time $t$, $\sum$ is the set of possible states (from the resource state model) and $t_0$ is the time resource $i$ was created. Constraints on state occupancy are bounds on the proportion of time spent in a particular state, or the ratio between the time spent in one state and time spent in one or more other states.

*K. Data Accuracy*

This metric captures the amount of error in (numerical) data supplied to or from a service, compared with a reference value from the thing the data is supposed to describe. The two main aspects of interest with this particular metric are the precision of the data (how close to the reference value is the data supposed to be) and the accuracy of the data (how close to the reference value the data is, compared to how close it was supposed to be). Subclasses of data accuracy may be needed to distinguish between different types of data used to describe the thing of interest (single values, arrays etc), and different ways of specifying precision (precision in terms of standard deviation, confidence limit etc), as well as to distinguish between things described by the data (e.g. aircraft landing times, fuel levels or prices). In the SERSCIS P-o-C, we are only really interested in the accuracy of predictions for the absolute time of future events, including the point when an aircraft will be available so turnaround can start (an input to the ground handler), the point when the aircraft will be ready to leave, and various milestones between these two points (e.g. the start and end of aircraft cleaning, etc). Constraints on accuracy are typically just upper bounds on the accuracy measure, e.g. accuracy should be less than 2.0. Such constraints apply individually to each data value relating to a given reference value.

*L. Data Precision*

This is a simple metric associated with the precision bands for data supplied to or from a service. Data that describes some reference value should always come with a specified precision, so measuring the precision is easy – one just has to check the precision as specified by whoever supplied the data. The reason it is useful to associate a metric with this is so one can specify constraints on data precision in SLA, to prevent data suppliers evading accuracy commitments by supplying data very poor (wide) precision bands. Subclasses of data precision are typically needed for different kinds of things described by data, and different sources of that data. For example, one might define different metrics to describe the precision in scheduled arrival times (taken from an airline timetable) and predicted arrival times (supplied by Air Traffic Control when the aircraft is en-route). Note that precision (unlike accuracy) is not a dimensionless number – it has the same units as the data it refers to, so metric subclasses should specify this. In the SERSCIS P-o-C testbed, it should be possible to ask a consumer task for the precision of data supplied to or by it. The response should ideally give the best, worst and latest precision estimates for the data corresponding to the metric. Constraints on data precision are simple bounds on its value. Typically they will appear in SLA, and define the worst-case precision that is acceptable to both parties. If data is provided with worse precision than this, the constraint is breached. This type of constraint is normally used as a conditional clause in compound constraint for data accuracy or accuracy distribution.

*M. Data Error*

This is a simple metric associated with the error in a data item relative to the reference value to which it relates. In some situations we may wish to specify and measure commitments for this 'raw' measure of accuracy, independently of its supposed precision. Subclasses of data error are typically needed for different kinds of things described by data, and different sources of data. In the SERSCIS P-o-C testbed, it should be possible to ask a consumer task for the error in data supplied to or by it once the reference value is known to the service. The response should ideally give the best, worst and latest error for data

 ISBN: 978-1-61208-109-0 

sent/received corresponding to the metric. Constraints on data error are simple bounds on its value. Typically they will appear in SLA, and define the worst-case error that is acceptable to both parties. If data is provided and turns out to have an error worse than this, the constraint is breached.

*N. Data Accuracy Compliance*

This metric captures the proportion of data items in a data set provided to or from a service whose accuracy is not worse than a specified limit. This metric is mathematically similar to the elapsed time compliance metric, and as before we may wish to use several accuracy compliance metrics for the same data at different accuracy levels, to approximate a data accuracy distribution function. Accuracy compliance can be measured by keeping track of the total number of data items, and how many of these had accuracy up to each specified level. The value of the metric is then the fraction of data items whose accuracy is within the specified level. In the SERSCIS P-o-C testbed, subclasses of accuracy compliance are typically used to distinguish between different accuracy levels, types of data and methods for defining precision, for data forecasting the time of events. To construct accuracy distributions it is necessary to classify those events so we know which forecasts to include in each distribution function. It should be possible to ask consumer tasks, services or service containers for the value of these compliance metrics. Constraints on accuracy compliance just specify bounds on the metric, thus specifying what proportion of data items can have accuracy worse than the corresponding accuracy limit.

*O. Auditable Properties*

Auditable property metrics are used to express whether a service satisfies some criterion that can't be measured, but can only be verified through an audit of the service implementation and behaviour. An auditable property will normally be asserted by the service provider, who may also provide proof in the form of accreditation based on previous audits in which this property was independently verified. Auditable properties are usually represented as State metrics: a state model is devised in which the desired property is associated with one or more states, which are related (out of band) to some audit and if necessary accreditation process. Subclasses are used to indicate different auditable properties and state models. Auditable property constraints typically denote restrictions on the resources (i.e. supplier services) used to provide the service. For example, they may specify that only in-house resources will be used, that staff will be security vetted, or that data backups will be held off site, etc. In SERSCIS, such terms are also referred to as Quality of Resourcing (QoR) terms. As with other state-based descriptions, auditable properties may be binary (true or false), or they may be ordered (e.g. to describe staff with different security clearance levels). It is also possible to treat Data Precision (and other data characteristics) as an auditable property which does not correspond to a state model.

## IV. RELATED WORK

Characterizing the performance of adaptive real-time systems is very difficult because it is difficult to predict the exact run-time workload of such systems. Transient and steady state behavior metrics of adaptive systems were initially drafted in [4], where the performance of an adaptive was evaluated by its response to a single variation in the application behavior that increased the risk of violating a performance requirement. A very simple set of metrics are used: *reaction time* which is the time difference between a critical variation and the compensating resource allocation, *recovery time* by which system performance returns to an acceptable level, and performance laxity which is the difference between the expected and actual performance after the system returns to a steady state. These metrics are further specialized in [1] by the introduction of *load profiles* to characterize the types of variation considered including *step-load* (instant) and *ramp-load* (linear) changes, and a *miss-ratio* metric which is the fraction of tasks submitted in a time window for which the system missed a completion deadline. System performance is characterized by a set of miss-ratio profiles with respect to transient and steady state profiles. A system is said to be stable in response to a load profile if the system output converges as the time goes to infinity, while transient profiles can measure responsiveness and efficiency when reacting to changes in run-time conditions. The SERSCIS-Ont metrics provide a superset of these concepts, appropriate to a wider range of situations where accuracy and reliability may be as important as performance and stability.

A more recent alternative approach to defining adaptive system metrics is given by [6,7]. Here the focus is on the system engineering concerns for adaptivity, and metrics are categorized into four types: *architectural* metrics which deal with the separation of concerns and architectural growth for adaptive systems [2], *structural* metrics which provide information about the role of adaptation in the overall functionality of a system (and vice versa), *interaction* metrics which measure the changes in user interactions imposed by adaptation, and *performance* metrics which deal with the impact of adaptation on system performance, such as its response time, performance latency, etc [2]. The focus of SERSCIS-Ont is to provide concrete and mathematically precise metrics covering performance and some aspects of interactivity, which can be used in such a wider engineering framework.

The most closely related work is found in the WSMO initiative [3], which has also formalized metrics for resource dependability. This was done with the intention of providing QoS aware service oriented infrastructures. Semantic SLA modeling using WSMO focuses principally on automated service mediation and on the service execution infrastructure [3]. By adding semantic descriptions for

 ISBN: 978-1-61208-109-0 

service parameters it is possible for agents to discover and rank services automatically by applying semantic reasoning. The WSMO initiative focused its modeling efforts on capturing service consumer requirements, which can then be used for service discovery. Work in [5] extends the WSMO ontology to include QoS and non-functional properties. This includes providing formal specifications for service level agreements including the units for measurement, price, CPU usage etc. However, the focus is still to support the description of services for orchestration purposes (service discovery and selection). SERSCIS-Ont is more even-handed. It can be used for service discovery and selection, but it is also designed to support service operators by introducing service protection measures from a provider's perspective such as the usage limits, service access and control decisions, as well as workflow adaption, etc.

SERSCIS-Ont is thus also related to the development and service management specifications such as WSDM. The WSDM-MOWS specification [9] defines 10 metrics which are used to measure the use and performance of a general Web Service. These include NumberOfRequests, NumberOfFailedRequests and NumberOfSucessfulRequests which count the messages received by the Web Service end point, and whether the service handles them successfully. In SERSCIS-Ont we have a more general Counter metric, of which these WSDM-MOWS metrics can be regarded as subclasses specifically for Web Service management. WSDM-MOWS also defines ServiceTime (the time taken by the Web Service to process all its requests), and MaxResponseTime and LatestResponseTime. In SERSCIS-Ont these would be modeled as subclasses of usage and elapsed time, and SERSCIS-Ont then provides additional metrics such as min/max/mean responses and response time compliance metrics. WSDM-MOWS specifies a state model for Web Service operation with states {UpState, DownState, IdleState, BusyState, StoppedState, CrashedState, SaturatedState}, and metrics CurrentOperationalState and LastOperationStateTransition all of which can be handled easily by SERSCIS-Ont. The one area where WSDM-MOWS goes beyond SERSCIS-Ont is in providing metrics for the size of Web Service request and response messages: MaxRequestSize, LastRequestSize and MaxResponseSize. These can be modeled with difficulty using SERSCIS-Ont usage metrics, but if SERSCIS-Ont were applied to Web Service management, some extensions would be desirable.

## V. VALIDATION EXPERIMENTS

To verify that SERSCIS-Ont really is applicable to the management of service performance and dependability, the project is conducting two types of experiments. Testbeds are being developed comprising SERSCIS dependability management tools along with emulated application services based on air-side operations at Vienna Airport. This will be a discrete event simulation in which realistic application-level requests and responses are produced, and the full (not emulated) management tools will be tested using SERSCIS-Ont metrics in service level agreements and monitoring and management policies.

Until the testbed is ready, SERSCIS validation work has focused on the use of stochastic process simulation based on queuing theory [10]. A simplified Markov chain model was developed for a single aircraft refueling service, and the resulting equations solved numerically to compute the expected behavior. This approach is faster and easier to interpret than a discrete event simulation, though it uses simpler and less realistic models of services and their interactions.

The basic model of the refueling service assumes that around 20 aircraft arrive per hour and need to be refueled. The service provider has 3 bowsers (fuel tankers) which can supply fuel to aircraft at a certain rate. The time taken for refueling varies randomly between aircraft depending on their needs and how much fuel they still have on landing, but the average time is 7.5 minutes. However, with only 3 bowsers, aircraft may have to wait until one becomes available before refueling can start. The SERSCIS-Ont metrics used to describe this service are:

- a counter metric for the number of aircraft refueled, and an associated usage rate metric for the number of aircraft refueled per hour;
- a non-recoverable usage rate metric for the time the bowsers spend actually refueling aircraft, from which we can also obtain the resource utilization percentage;
- an elapsed time metric for the amount of time spent by aircraft waiting for a bowser (the refueling service can't control how long the refueling takes, so QoS is defined in terms of the waiting time only); and
- elapsed time compliance metrics for the proportion of aircraft that have to wait for different lengths of time between 0 and 20 minutes.

We also assume that the service will refuse an aircraft, i.e. tell it to use another refueling company rather than wait, if it would become the $10^{th}$ aircraft in the queue. This is captured by a further counter metric, which is used to find the proportion of arriving aircraft that are refused service.

The first simulation considered an unmanaged service (no SLAs), and produced the following behavior (See Table 1):

TABLE 1: UNMANAGED SERVICE SIMULATION

| Metric | Value |
|---|---|
| Service load | 20 aircraft / hour |
| Service throughput | 19.5 aircraft / hour |
| Percentage of aircraft that don't have to wait | 33.6% |
| Percentage that don't have to wait more than 10 mins | 74.6% |
| Percentage that don't have to wait more than 20 mins | 94.4% |
| Percentage of aircraft refused service | 2.6% |
| Mean waiting time | 6.1 mins |
| Resource utilization | 81.2% |

 ISBN: 978-1-61208-109-0 

The QoS is relatively poor because the random variation in aircraft arrival and refueling times means queues can build up, leading to a high proportion of aircraft having to wait, and some having to wait for a long time or even being sent to other service providers.

To investigate how the metrics could be used to manage the service, the simulation was extended so airlines must have an SLA with the service provider before they can use the service. Each SLA lasts on average 1 week, and allows an airline to refuel an average of 3 aircraft per hour. The extended model assumed about one new SLA per day would be signed, giving an average load roughly similar to the total load in the first simulation. We also assumed the service provider would refuse to agree more than 12 SLA at a time, so the load could temporarily rise up to 50% higher than the capacity of its resources. We wished to investigate how well the use of SLA as a pre-requisite for service access allowed such overloads to be managed. The results of this second simulation were as follows (See Table 2):

TABLE 2: MANAGED SERVICE SIMULATION

| Metric | Value |
|---|---|
| Service load | 0-36 aircraft / hour |
| Service throughput | 21.1 aircraft / hour |
| Percentage of aircraft that don't have to wait | 22.4% |
| Percentage that don't have to wait more than 10 mins | 60.4% |
| Percentage that don't have to wait more than 20 mins | 89.7% |
| Percentage of aircraft refused service | 4.9% |
| Mean waiting time | 9.4 mins |
| Resource utilization | 87.8% |

While the use of this SLA allowed the service provider to anticipate the load from a pool of potential consumers, it couldn't improve QoS with a fixed set of resources. In fact, the compliance metrics are now much worse than before, with only a small increase in the total throughput because the load exceeds the resource capacity around 25% of the time. Further tests showed that reducing the number of SLA the service accepts doesn't help much as this only lowers the long term average load, whereas overloads and long queues arise from shorter-term fluctuations. The limit would have to be much lower (and the throughput substantially lower) before the compliance metrics were good enough to be of interest to customers.

The final experiment used a different type of SLA in which each customer can still have 3 aircraft serviced per hour on average, but only one at a time. To handle this, we used a non-recoverable usage rate metric for the number of aircraft in the system and specified in the SLA that this could not exceed 1. This simulation produced the following (See Table 3):

TABLE 3: CONSTRAINED SLA SERVICE SIMULATION

| Metric | Value |
|---|---|
| Service load | 0-36 aircraft / hour |
| Service throughput | 17.9 aircraft / hour |
| Percentage of aircraft that don't have to wait | 50.6% |
| Percentage that don't have to wait more than 10 mins | 96.0% |
| Percentage that don't have to wait more than 20 mins | 99.9% |
| Percentage of aircraft refused service | 0% |
| Mean waiting time | 3.4 mins |
| Resource utilization | 74.7% |

Evidently, if this last type of SLA were enforced by a suitable management procedure, it would allow the service to protect itself from overloads, without a huge drop in the service throughput. Further experiments showed that if the permitted long-term load per SLA were pushed up to 3.5 aircraft per hour, the throughput would reach 19.7 aircraft per hour (more than the original unmanaged service), yet the compliance metrics would stay above 90%. This provides a good indication that the SERSCIS-Ont metrics can be used to describe service management and protection constraints, as well as consumer QoS measurements and guarantees.

## VI. CONCLUSIONS

This paper describes a base metric model that provides a uniform abstraction for describing service behavior in an adaptive environment. Such an abstraction allows services to be composed into value chains, in which consumers and providers understand and can manage their use of services according to these metrics.

A service provider, having analyzed the application service that it is offering, defines a metric ontology to describe measurements of the relevant service behavior. This ontology should refer to the SERSCIS base ontology, and provide subclasses of the base metrics to describe each relevant aspect of service behavior. Note that while each service provider can in principle define their own metrics ontology, it is may be advantageous to establish 'standard' ontologies in particular domains – this reduces the need for translation of reported QoS as it crosses organizational boundaries.

Validation simulations provide a good indication that the SERSCIS-Ont metrics are useful for describing both service management and protection constraints, and service dependability and QoS guarantees.

 ISBN: 978-1-61208-109-0 

## ACKNOWLEDGMENT

The research leading to these results has received funding from the European Community's Seventh Framework Programme under grant agreement no. 225336, SERSCIS.

## REFERENCES

[1] C. Lu, J.A. Stankovic, T.F. Abdelzaher,G.Tao, S.H. Son and M.Marley, "Performance Specifications and Metrics for Adaptive Real-Time Systems,"In Real-Time Systems Symposium 2000.

[2] C. Raibulet and L. Masciadri. "Evaluation of Dynamic Adaptivity Through Metrics: an Achievable Target?". In the paper proceedings of the 8th working IEEE/IFIP Conference on Software Architecture. WICSA 2009.

[3] D.Roman, U.Keller, H. Lausen, R.L.J. de Bruijn, M. Stolberg, A.Polleres, C.Feier, C.Bussler and D.Fensel. "Web service modelling ontology". Applied Ontology. I (1):77-106, 2005.

[4] D. Rosu, K. Schwan, S. Yalamanchili and R. Jha, "On Adaptive Resource Allocation for Complex Real-Time Applications," 18th IEEE Real-Time Systems Symposium, Dec., 1997. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[5] I. Toma, D. Foxvog, and M.C. Jaeger. "Modelling QoS characterists in WSMO". In: Proceedings of the 1st workshop on Middleware for Service Oriented Computing. November 27-December 01, 2006.

[6] L. Masciadri, "A Design and Evaluation Framework for Adaptive Systems", MsC Thesis, University of Milano-Bicocca, Italy, 2009.

[7] L. Masciadri, and C. Raibulet, "Frameworks for the Development of Adaptive Systems: Evaluation of Their Adaptability Feature Software Metrics", Proceedings of the 4th International Conference on Software Engineering Advances, 2009, in press.

[8] Collected Works of John von Neumann, 6 Volumes. Pergamon Press, 1963.

[9] WSDM-MOWS Specificaiton. www: http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.htm (Last accessed Aug 2010).

[10] D. Gross and C.M. Harris. Fundamentals of Queueing Theory. Wiley, 1998.

 ISBN: 978-1-61208-109-0 

# Attainable Capacity Aware Routing Metric for Wireless Mesh Networks

Nemesio A. Macabale Jr.[1,2], Roel M. Ocampo[2] , Cedric Angelo M. Festin[3]
namacabale@up.edu.ph, roel@eee.up.edu.ph, cedric@cs.up.edu.ph

[1]Dept of Information Technology
Central Luzon State University
Science City of Munoz, Philippines

[2]Electrical and Electronics Engineering Institute
[3]Dept of Computer Science
University of the Philippines – Diliman
Quezon City, Philippines

***Abstract*— We propose a new metric, called the attainable capacity (ACAP) aware routing metric, to address issues on throughput, interference and load imbalance in wireless mesh networks. In contrast with previously proposed metrics, ACAP takes into account the busyness and shared nature of the wireless channel together with the combined effect of the transmission rates of the nodes that share it. Accordingly, paths with highest attainable capacity are chosen as the best paths. In the process, regions with higher degree of congestion are also avoided to improve throughput and load distribution across the network. We present an analysis of the above interaction and uses it to define ACAP. We expect ACAP to be significantly better than existing metrics in discriminating congested regions and in finding higher capacity routes. Like many of these existing metrics, APAC is suitable to both multi-radio and multi-channel wireless mesh networks.**

***Keywords - wireless mesh networks, routing, routing metric, congestion awareness, attainable capacity***

## I. Introduction

Wireless mesh networks (WMN) have emerged to be a cheaper and flexible alternative for quick deployment of wireless services for a large variety of applications such as broadband home networking and automation [1] , community and neighborhood networking [2], transportation systems [3], spontaneous (emergency/disaster) networking[4], and others.

However, despite advances in this field, many research challenges remain [5]. One issue is the use of routing metrics that do not scale well, such as hop count: network throughput drops significantly as the number of nodes or hops increases in the network [6]. This paper proposes the attainable capacity (ACAP) aware routing metric to address issues on throughput, interference,load imbalance, and congestion problems as networks grow.

The rest of the paper is organized as follows: Section II discusses the motivation and related work, while Section III presents an analysis that lead to the design of ACAP. Section IV discusses ACAP's implementation details. Finally, Section V concludes.

## II. Motivation and Related Work

In both wired and wireless networks, a routing algorithm's function is to discover a path for a packet to traverse from its source to its destination. A routing algorithm in return may find more than one route. To decide which is best, it uses routing metrics [7]. In a multi-hop WMN, routing becomes more critical than in wired networks, because the wireless medium is shared and is highly dynamic [8]. Different packet flows may interfere with each other even when they do not necessarily traverse the same path, consequently congesting that direction thus lowering throughput significantly.

The simplest and most commonly-used routing metric in WMN is the hop-count metric, as used in DSR [9] , AODV [10], and DSDV[11]. It reflects the path- length in hops, and in many cases the shortest physical path is used. However, it is insensitive to the quality of links between hops and to the degree of congestion on the link [12].

ETX [12] and ETT [13] on the other hand are able to measure quality of links but not congestion. Some load balancing protocols like WLAR [14] and DLAR [15] can avoid loaded nodes, but cannot determine link capacities.

Among the metrics that are similar to ACAP in functionality, ALARM [16] identifies paths with better capacity and nodes with less load. However, it cannot measure interfering transmissions from neighboring nodes. Thus, it may avoid loaded nodes but not necessarily congested regions of the network. The ILA routing metric [17] claims to have solved this issue. However, its metric measures congestion in terms of the average load of nodes within a collision domain (see Fig. 1). In this case, it will not be able to distinguish between regions having more interfering nodes over regions having less, as long as the average loads are the same. Thus, it can not determine capacity accurately since a domain with less nodes that share a channel can provide higher achievable data rates. ACAP, on the other hand, solves these issues by estimating the achievable capacity in proportion to the load of the channel, and in conjunction with the transmission rates (minus packet losses) of the individual nodes that share it.

## III. Design of the Proposed Metric

We begin our analysis by looking at a node *j*'s collision domain. It is comprised of *j's* neighbor nodes within its carrier sensing range that operate on the same channel. All transmissions of these nodes interfere with that of *j*'s. See Fig. 1, assuming circular carrier sensing range.

Because of the shared nature of the channel within *j's* collision domain, the capacity of the link that may be achieved in choosing *j* as a next hop node depends on the activities happening in the channel. If the channel is idle, the achievable capacity is close to the full capacity of the link, less overhead and packet losses. If there are activities in the channel, then the link's achievable capacity is less.

### A. Channel Busyness and Utilization

In determining the ACAP of the link in considering a node *j* as the next-hop from a node *i*, we quantify the degree of *busyness* of the channel within *i* and *j's* collision domain. Although other authors define the degree of busyness as the fraction of time the channel is inferred to be busy [18], in our work, we refine this definition of busyness to pertain to the fraction of time the channel is *sensed* (rather than inferred) to be busy.

Busyness, as used in our work, is different from channel utilization. Channel utilization is typically defined as the achieved throughput related to the capacity of the communication medium [7]

 ISBN: 978-1-61208-109-0 

[16], which could achieve a maximum value less than 100%, because of overhead, packet losses, and queuing issues. On the other hand, channel busyness is defined as the fraction within a given period when transmission occurs over the wireless medium, which may attain a maximum value of 100%.

### B. The Attainable Capacity Aware (ACAP) Routing Metric

In a simulation study, it is observed that channel busyness increases linearly with aggregate input traffic until channel saturates [18]. When the channel is saturated, although input traffic increases, throughput does not increase because all time slots are utilized. In [19], a channel saturates when the aggregate input traffic reaches 80% of the nominal bit rate. In a separate study [20] that looks onto the saturation throughput between a 802.11b access point and a laptop, similar behavior was observed except that the saturation throughput reached only 50 to 70% of the channel capacity (5 different wireless brands were individually tested). In these studies, the input traffic approximates the throughput linearly until saturation. At saturation, as more input traffic is injected, the throughput either hovers around a constant value [18] or asymptotically drops to a lower one [18]. In the context of this work, the studies imply two things. First, that the rated capacity of a channel is never reached even if there is only a pair of communicating nodes, because of packet losses, overheads, and delays. Secondly, it can be said that the capacity that a channel may offer decreases linearly until saturation. The second implication is summarized in Fig. 2. Using methods from analytic geometry [21], the ACAP in connecting to *j* is derived as:

$$ACAP(j) = ACAP(j)_{idl} - CB(j)(ACAP(j)_{idl} - ACAP(j)_{sat}) \quad (1)$$

where:
$ACAP(j)$ is the attainable capacity (ACAP) in connecting to $j$
$ACAP(j)_{idl}$ is the ACAP when the channel is idle
$ACAP(j)_{sat}$ is the ACAP when the channel is saturated
$CB(j)$ is the degree of channel busyness of node $j$'s collision domain

ACAP(*i*) is obtained following the same analysis. In turn, the ACAP of the link k between nodes *i* and *j* is defined as:

$$ACAP(k) = \frac{1}{\frac{1}{ACAP(i)} + \frac{1}{ACAP(j)}} \quad (2)$$

where:
$ACAP(k)$ is the attainable capacity of link between nodes $i$ and $j$
$ACAP(j)$ is the attainable capacity in connecting to $j$
$ACAP(k)$ is the attainable capacity in connecting to $i$

Consequently the attainable capacity of the path between a pair of source and destination is given in 3. If the routing algorithm finds more than one path, it selects the path with the highest ACAP metric.

$$ACAP(P) = \frac{1}{\sum_{k \in P} \frac{1}{ACAP(k)}} \quad (3)$$

where:
$ACAP(P)$ is the attainable capacity on path $P$
$ACAP(k)$ is the actual attainable capacity of link $k$
$k$ is a link on path $P$

We do not make any assumption about the operating channel of a collision domain; we only require to consider *i* and *j's* collision domains to operate on the same channel as that of link k. If some collision domains operate over different channels, the analysis would follow the same process. Therefore, (2) can determine the ACAP of a link and (3) can determine ACAP of a path. Thus, ACAP is suitable for multi-radio and multi-channel mesh networks. At the moment, our analysis assumes negligible channel switching cost.

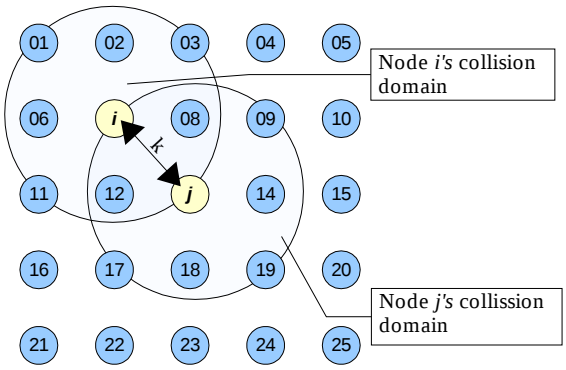



*Figure 1: A Wireless Mesh Network with 25 nodes*

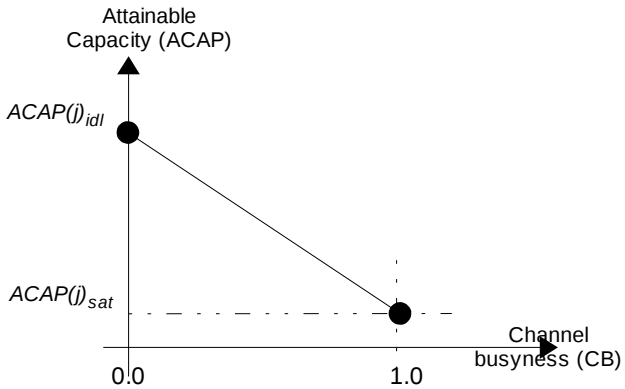



*Figure 2: Attainable Capacity (ACAP) vs. Channel busyness (CB)*

## IV. Implementation

In this section, we define both the attainable capacity and the channel busyness when the channel is idle and when it is saturated. We assume that all nodes in a collision domain have a packet to transmit at saturation. In doing so, we use 802.11 equipped nodes operating using the basic access mode of the distributed communication function (DCF) of the 802.11 [22].

### A. The value for the ACAP when the channel is idle (ACAPidl)

When the channel is idle, the ACAP in choosing *j* as the next-hop node can never be more than, and usually is less than, the nominal bit rate of the link between *i* and *j* due to overheads and packet losses [19],[18],[20]. Discounting overheads, but considering packet losses, the ACAP of the link k connecting *i* and *j* when the channel is idle can be defined as:

$$ACAP(k)_{idl} = p_{ij} p_{ji} r_{ij} \quad (4)$$

where:
$ACAP(k)_{idl}$ is the attainable capacity when the channel is idle
$p_{ij}$ and $p_{ji}$ are the forward and reverse packet delivery ratios
$r_{ij}$ is the nominal bit rate between $i$ and $j$

Here, we do not include the overhead in the definition of the ACAP because we are not trying to come up with an accurate value for the attainable capacity that a path may provide. Rather, we would like to compute relative values that may be compared

between candidate routing paths. The overhead is common to other links and so its effect, hence, the omission is justified.

The use of forward and reverse packet delivery ratios is due to channel asymmetry and to estimate them periodic probe packets can be used [12],[13].

### B. The value for the ACAP when the channel saturates

At channel saturation, the 802.11 distributed communication function (DCF) provides equal transmission opportunities among nodes in a collision domain [23]. Following the analysis made in [24],[25], and again discounting overhead but considering packet losses, this value is defined as:

$$r(j)_{sat} = \frac{1}{\sum_{n \in D_j \wedge n \neq j} \frac{1}{p_{nj} p_{jn} r_{nj}}} \quad (5)$$

*where*:
$r(j)_{sat}$ *is the channel's saturation data rate within* $j$*'s collision domain*
$D_j$ *is the set of nodes within* $j$*'s collision domain*
$n$ *is a node within* $j$*'s collision domain*
$p_{ij}$ and $p_{ji}$ *are the forward* and *reverse packet delivery ratios*
$r_{nj}$ *is the nominal bit rate of node* $n$ in *connecting* to *node* $j$
$j$ *is the node being considered*

Rewriting (5) in terms of ACAP, $ACAP(j)_{sat}$ is defined in (6). $ACAP(i)_{sat}$ should be obtained in the same manner.

$$ACAP(j)_{sat} = \frac{1}{\sum_{n \in D_j \wedge n \neq j} \frac{1}{ACAP(nj)_{idl}}} \quad (6)$$

*where*:
$ACAP(j)_{sat}$ *is* $j$*'s collision domain attainable capacity at saturation*
$r(j)_{sat}$ *is the channel's saturation data rate*
$D_j$ *is the set of nodes within* $j$*'s collision domain*
$n$ *is a node within* $j$*'s collision domain*
$ACAP(nj)_{idl} = p_{nj} p_{jn} r_{nj}$

However, it should be noted that (6) gives an ACAP at saturation that does not take into account the effect of collision domains of nodes outside the path but whose range overlaps with those of the nodes along the path. We presume that extending the definition to include their effect will increase the complexity of finding the solution. We opted to initially have a workable one, as given by (6) and tackle such scenario as part of a future work.

### C. The channel busyness (CB)

Channel busyness in a collision domain is defined as the fraction of time within a given period where the channel is being used for transmission. A previous definition was presented in [18] that seems simpler to implement, because it is obtained from just observing the collision probability within the channel as opposed to actually monitoring the channel for a given period (that we intend to do). However, this may not be accurate since the DCF of 802.11 actually avoids collision through its back-off mechanism, and this value is kept at minimum until saturation point. Nevertheless, their result showed some accuracy even in the presence of other causes of packet loss (like fading). We are currently validating this, as the simplicity of their definition might prove useful in this work.

We intend to monitor activity on the channel through carrier sensing. A node determines the channel as busy when a node (not necessarily the sensing node) is sending a message to another node, or the sensing node itself is actually transmitting data, whether transmission is successful or not. For a given period, (equal to 100ms synchronized with the transmission of the 802.11 beacon frame [22]), the degree of busyness from the perspective of a node $j$ is defined in (7).

$$CB_j = \frac{\sum T_{busy} + \sum T_{transmitting}}{\sum T_{idle} + \sum T_{busy} + \sum T_{transmitting}} \quad (7)$$

*where*:
$CB_j$ *is the channel busyness ratio* from *within* $j$*'s collision domain*
$\sum T_{idle}$ *is the duration that node* $j$ *senses the channel as idle*
$\sum T_{busy}$ *is the duration that node* $j$ *senses the channel as busy*
$\sum T_{transmitting}$ *is the duration that node* $j$ *is transmitting frames*

Here $T_{idle}$, $T_{busy}$ *and* $T_{transmitting}$ are computed as follows [22],[18]:
$T_{idle}$ = number of slot-time sensed as idle * 20 µs
$T_{busy}$ = number of slot-time sensed as busy * 20 µs
$T_{transmitting} = T_{successful} + T_{collision}$
$T_{successful}$ = data/bitrate + ack/bitrate + SIFS + DIFS + 2* $T_{pr}$
$T_{collision}$ = data/bitrate + ack-timeout + SIFS + DIFS + 2* $T_{pr}$

Based on 802.11b[22] values and without using RTS/CTS mechanism, the constants above are slottime = 20µs; ack = 14-byte frame; ack-time out = 22µs, (as used in [26]); SIFS (short interframe sequence) = 10 µs; PIFS (point interframe sequence) = SIFS + slottime = 30 µs; DIFS (distributed interframe sequence) = SIFS + 2 x slottime = 50 µs; $T_{pr}$ = 96 µs, is the PLCP (Physical Layer Convergence Protocol) preamble and header. The short PLCP is put here, since it is used for most available nominal bit rates in 802.11

DIFS is the amount of time a station must sense a clear radio before beginning a new transmission sequence. SIFS is the amount of time a station must wait before sending or beginning to receive a ACK frame, RTS, or CTS. $T_{pr}$ is the time occupied by the PLCP header introduced by the physical layer for mapping MPDU (mac protocol data unit) into a suitable PDU (Physical Data Unit)[23].

However, we need to smoothen the impact of the sudden changes in traffic. We employ a moving average for the channel busyness using a tunable parameter α. To make it simple, we will initially use 0.5. The channel busyness is, thus, defined as:

$$CB_j(t) = (1-\alpha) \times CB_j(t-1) + \alpha \times CB_j \quad (8)$$

*where*:
$CB_j(t)$ *is the current value of the moving average of the channel busyness ratio within* $j$*'s collision domain*
$\alpha$ *is a tunable parameter*: $0 \leq \alpha \leq 1$, *here* 0.5 *is used*
$CB_j$ *is the current computed channel busyness*
$CB_j(t-1)$ *is the previous smoothed average channel busyness*
$t$ *refers* to *the current measuring period*

### D. ACAP Summary

ACAP is a measure of the attainable capacity of a link based on the shared nature and busyness of a channel. The more busy a channel, the less it is capable of accepting input traffic without dropping packets. Its shared nature, on the other hand, is affected by the quantity of nodes sharing the channel and their respective transmission rates. The more nodes that share a channel, the less share a node gets on the channel capacity. Further, in a self-configuring WMN that implements a distributed channel access mechanism, similar to 802.11's DCF, the node with the lowest nominal-bit-rate penalizes the high-bit-rate ones: at saturation, each node gets a bit rate that is no more than the lowest bit rate [27]. Put together, using the analytical model presented above, the ACAP of a

 ISBN: 978-1-61208-109-0 

link is obtained. The ACAP of the path becomes the sum of the ACAP's of the links that comprise the path. This value is then used as basis for choosing the best path among candidate paths.

ACAP is suitable for both multi-radio and multi-channel wireless mesh networks as pointed out earlier. We compute the ACAP one link at a time, per collision domain.

## V. Conclusion and Future Work

We proposed a new routing metric called the Attainable Capacity Aware routing metric (ACAP) for wireless mesh networks. ACAP estimates the attainable capacity of a link based on the shared nature and busyness of a channel within a link's end-nodes' collision domain. The shared nature of the channel is affected by the quantity of nodes within the collision domain and their respective transmission rates. The quality of the links based on packet losses has also been incorporated into the ACAP metric

We expect ACAP to perform better than recently proposed routing metrics, such as ALARM and ILA, because it can better discriminate congestion and accurately estimate capacity by incorporating the busyness and shared nature of the channel, and in conjunction with the quantity of and respective nominal bit rates (minus packet loss) of contending nodes.

Our current work focuses on testing our metric in a simulation set-up similar to [19] and [17] then implementing the experiment in actual test-bed whose set-up is similar to [28].

In the future, we intend to include a broader framework that takes into consideration the effect of overlapping domains. Other mechanisms to account for channel busyness will also be explored .

## Acknowledgment

This work has been supported by the Engineering Research and Development for Technology (ERDT) Consortium, Department of Science and Technology – Science Education Institute (DOST-SEI), Republic of the Philippines.

## References

[1] 1TouchMovie.com, “ZigBee, Insteon, Z-Wave, and inifiNET Wireless Mesh Networks for Home Automation,” Availabe at: http://www.1touchmovie.com/wireless_mesh_networks.html. Access on: March 23. 2010

[2] Riverfront Apartments, “Riverfron Apartments”, Available at: http://meraki.com/solutions/cs/riverfront/.” Accessed on, Mar 23, 2010

[3] V. Naumov and T. Gross, “Connectivity-Aware Routing (CAR) in Vehicular Ad-hoc Networks,” *in Proc. 26th IEEE International Conference on Computer Communications*, IEEE, 2007.

[4] R. Knopp, N. Nikaein, C. Bonnet, H. Aiache, V. Conan, S. Masson, G. Guib, and C. Le Martret, “Overview of the Widens Architecture, A Wireless Ad Hoc Network for Public Safety,” 2004.

[5] I.F. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: a survey,” *Comput. Netw. ISDN Syst.*, vol. 47, 2005, pp. 445-487.

[6] S. Srivathsan, N. Balakrishnan, and S.S. Iyengar, “Scalability in Wireless Mesh Networks,” *Guide to Wireless Mesh Networks*, 2009, pp. 325-347.

[7] A.S. Tanenbaum, *Computer Networks*, Prentice Hall, 2002.

[8] T. Rappaport, *Wireless Communications: Principles and Practice (2nd Edition)*, Prentice Hall PTR, 2002.

[9] David B. Johnson, David A. Maltz, and Yih-Chun Hu, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR), Internet-Draft,” Jul. 2004.

[10] C. Perkins, E. Belding-Royer, and S. Das, “Ad hoc On-demand Distance Vector (AODV) Routing, rfc 4561,” Jul. 2003.

[11] C.E. Perkins and P. Bhagwat, “Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers,” *SIGCOMM Comput. Commun. Rev.*, vol. 24, 1994, pp. 234-244.

[12] D.S.J.D. Couto, D. Aguayo, J. Bicket, and R. Morris, “A high-throughput path metric for multi-hop wireless routing,” *Proceedings of the 9th annual international conference on Mobile computing and networking*, San Diego, CA, USA: ACM, 2003, pp. 134-146.

[13] R. Draves, J. Padhye, and B. Zill, “Routing in multi-radio, multi-hop wireless mesh networks,” *Proceedings of the 10th annual international conference on Mobile computing and networking*, Philadelphia, PA, USA: ACM, 2004, pp. 114-128.

[14] D. Choi, J. Jung, K.Y. Kwon, D. Montgomery, and H. Kahng, “Design and Simulation Result of a Weighted Load Aware Routing (WLAR) Protocol in Mobile Ad Hoc Network,” *Information Networking*, 2005, pp. 178-187.

[15] S. Lee and M. Gerla, “Dynamic load-aware routing in ad hoc networks,” *Communications, 2001. ICC 2001. IEEE International Conference on*, 2001, pp. 3206-3210 vol.10.

[16] A.A. Pirzada, R. Wishart, M. Portmann, and Jadwiga Indulska, “ALARM: An Adaptive Load-Aware Routing Metric for Hybrid Wireless Mesh Networks,” *in Proc. 32nd Australasian Computer Science Conference*, Wellington, New Zealand: 2009.

[17] D.M. Shila and T. Anjali, “Load aware traffic engineering for mesh networks,” *Comput. Commun.*, vol. 31, 2008, pp. 1460-1469.

[18] H. Zhai, X. Chen, and Y. Fang, “How Well Can the IEEE 802.11 Wireless LAN Support Quality of Service,” *IEEE Transactions on Wireless Communications*, vol. 4, Nov. 2005, pp. 3084 - 3094.

[19] L.T. Nguyen, R. Beuran, and Y. Shinoda, “A load-aware routing metric for wireless mesh networks,” *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on*, 2008, pp. 429-435.

[20] E. Pelletta and H. Velayos, “Performance Measurements of the Saturation Throughput in IEEE 802.11 Access Points,” *Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05)*, Riva del Garda, Trentino, Italy: , pp. 129-138.

[21] A. Robson, *Introduction to Analytical Geometry*, Cambridge University Press, 2009.

[22] IEEE, “802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications- Revision of the 802.11-1999 standard,” 2007.

[23] B.O.A. Petrick, *The IEEE 802.11 Handbook: A Designer's Companion*, Institute of Electrical & Electronics Enginee, 2005.

[24] O. Ekici and A. Yongacoglu, “A novel association algorithm for congestion relief in IEEE 802.11 WLANs,” *Proceedings of the 2006 international conference on Wireless communications and mobile computing*, Vancouver, British Columbia, Canada: ACM, 2006, pp. 725-730.

[25] V. Mhatre, H. Lundgren, and C. Diot, “MAC-aware routing in wireless mesh networks,” *In Proc. Fourth Annual Conference on Wireless on Demand Network Systems and Services*, Oberguyrgl: IEEE, 2007.

[26] Air-Stream Community Wireless Network, "Changing ACK timeout on various Oses.” Available at: http://www.air-stream.org.au/Change_ACK, accessed on: April 10, 2010

[27] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda, “Performance Anomaly of 802.11b,” *In Proc Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, IEEE, 2003, pp. 836-843.

[28] D. Johnson and G. Hancke, “Comparison of two routing metrics in OLSR on a grid based mesh network,” Ad Hoc Netw., vol. 7, 2009, pp. 374-387.

 ISBN: 978-1-61208-109-0 

# Ocean Vessel Trajectory Estimation and Prediction Based on Extended Kalman Filter

Lokukaluge P. Perera
Centre for Marine Technology and Engineering
Technical University of Lisbon, Instituto Superior Técnico, Lisbon, Portugal
prasad.perera@mar.ist.utl.pt

Carlos Guedes Soares
Centre for Marine Technology and Engineering
Technical University of Lisbon, Instituto Superior Técnico, Lisbon, Portugal
guedess@mar.ist.utl.pt

*Abstract* — **The accurate estimation and prediction of the trajectories of maneuvering vessels in ocean navigation are important tools to improve maritime safety and security. Therefore, many conventional ocean navigation systems and Vessel Traffic Management & Reporting Services are equipped with Radar facilities for this purpose. However, the accuracy of the predictions of maneuvering trajectories of vessels depends mainly on the goodness of estimation of vessel position, velocity and acceleration. Hence, this study presents a maneuvering ocean vessel model based on a curvilinear motion model with the measurements based on a linear position model for the same purpose. Furthermore, the system states and measurements models associated with a white Gaussian noise are also assumed. The Extended Kalman Filter is proposed as an adaptive filter algorithm for the estimation of position, velocity and acceleration that are used for prediction of maneuvering ocean vessel trajectory. Finally, the proposed models are implemented and successful computational results are obtained with respect to prediction of maneuvering trajectories of vessels in ocean navigation in this study.**

**Keywords- Trajectory estimation; Trajectory prediction; Target tracking; Extended Kalman Filter; Curvilinear motion model.**

## I. INTRODUCTION

The European Union (EU) is surrounded by a busy and complex set of sea routes. Furthermore, over 90% of EU external trade transports by the sea and over 3.7 billion tones of freight per year are transferred through the EU ports. In addition, passenger traffic in the seas around the regions of the EU is presently approximated to 350 million passenger journeys per year [1]. With the increased demand for maritime transportation of passengers and freight, the increase maritime safety and security issues are highlighted in this region. Therefore, the proposal for local community vessel traffic monitoring and information systems has been considered by the EU Directive 2002/59 [2] for highly dense maritime traffic regions to equip with the regional Vessel Traffic Monitoring & Reporting (VTMR) systems to improve the safety and security.

The detection, tracking, trajectory estimation and trajectory prediction of maneuvering vessels are important facilities for navigation systems as well as the VTMR systems to improve safety, security and survivability in ocean navigation. However, conventional ocean navigation and VTMR systems are equipped with several marine instruments for the same purpose: Radar, Laser, Automatic Radar Plotting Aid (ARPA), and Automatic Identification System (AIS). Even though the first experimental Radar systems were envisioned, around 1920, for ship collisions avoidance [3], advanced Radar facilities were developed for the land and air navigation systems in later stages. Furthermore, Laser systems are proposed by recent studies [4] that will be important part of the target detection in close proximity.

ARPA provides accurate information of range and bearing of nearby vessels. AIS is capable of giving all the information on vessel structural data, position, course, and speed. The AIS simulator and marine traffic simulator have been implemented on several experimental platforms to perform navigation safety and security studies [5]. However, there are many challenges faced by the ocean surveillance [6]: The larger surveillance volume, synchronization of targets and sensors, noisy signal propagation environment and multi-target situation observations.

Furthermore, the effective estimation and prediction of trajectories of maneuvering ocean vessels have not been facilitated with the present navigation and VTMR systems. Therefore, main objective in this study is to propose a methodology for the navigation and VTMR system to estimate the present position, velocity, and acceleration of the vessels by observing or measuring only the noisy vessel positions. Furthermore, the estimated position, velocity, and acceleration can be used to predict the future navigation trajectories of the ocean vessel, which is another advantage of this proposed study.

However, the effective prediction of maneuvering trajectories of ocean vessels depends on the accuracy of data that are extracted from observations of the positions from the respective ocean vessels and the adaptive capabilities of the estimation algorithm. Therefore, accurate instruments with low sensor noise, as well as the capable optimal/sub-optimal adaptive estimation algorithm, should be formulated to archive accurate prediction in ocean vessel navigation. Several methods for the estimation and prediction of the maneuvering trajectories have been proposed by recent studies with respect to the land, air and ocean navigation systems. However, almost all the target tracking methods that are used for trajectory estimation and prediction are model based with respect to the recent studies [7].

 ISBN: 978-1-61208-109-0 

The models that are used in the estimation and prediction of maneuvering target tracking models can be grouped into two general categories: Continuous-time and Discrete-time models. The continuous-time model in maneuvering target tracking that includes the dynamic system model as well as the measurement model can be formulated as [7]:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f\left(\mathbf{x}(t), \mathbf{u}(t)\right) + \mathbf{w}_x(t) \\ \mathbf{z}(t) &= f\left(\mathbf{x}(t)\right) + \mathbf{w}_z(t) \end{aligned} \tag{1}$$

where $\mathbf{x}(t)$, $\mathbf{u}(t)$ and $\mathbf{z}(t)$ represent the system states, control inputs and measurements in continuous-time, respectively. Furthermore, $\mathbf{w}_x(t)$ and $\mathbf{w}_z(t)$ are the process and measurement noise of the system in continuous-time, respectively. The discrete-time model in maneuvering target tracking that includes the dynamic system model as well as the measurement model can be formulated as [7]:

$$\begin{aligned} \dot{\mathbf{x}}(k+1) &= f\left(\mathbf{x}(k), \mathbf{u}(k)\right) + \mathbf{w}_x(k) \\ \mathbf{z}(k) &= f\left(\mathbf{x}(k)\right) + \mathbf{w}_z(k) \end{aligned} \tag{2}$$

where $\mathbf{x}(k)$, $\mathbf{u}(k)$ and $\mathbf{z}(k)$ represent the system states, control inputs and measurements in discrete-time, respectively. Furthermore, $\mathbf{w}_x(k)$ and $\mathbf{w}_z(k)$ are the process and measurement noise of the system in discrete-time, respectively.

However, combination of continuous-time and discrete-time models in maneuvering target tracking approaches are also been used in recent studies. A nonlinear kinematic system model associated with a white Gaussian state noise for the ocean vessel in considered in this study as further described in Section III.

Furthermore, a linear model for the measurement system associated with a white Gaussian measurement noise is also considered in this study. An estimation algorithm of the Extended Kalman Filter (EKF) is proposed for the prediction of the future navigational trajectories of ocean vessels. However, the EKF algorithm is working in this study as an adaptive filter that estimates the system states of position, velocity and acceleration.

The work presented in this study is part of the on-going effort to formulate an intelligent collision avoidance system in ocean navigation, described in [8] and [9]. The organization of this paper is as follows: The recent developments in the Detection and Tracking of Moving Objects are discussed in Section II. The detail view of the Estimation and Prediction of Ocean Vessel Trajectories are presented in the Section III. Finally, the Computational Simulations and Conclusion are presented in Section IV and V, respectively.

## II. RECENT DEVELOPMENTS IN DETRECTION AND TRACKING

The Detection and Tracking of Moving Objects (DTMO) and Simultaneous Localization and Mapping (SLAM) are important divisions that are developed under the autonomous navigation systems. Hence, the tools developed under the DTMO and SLAM can be adopted for the estimation and prediction of maneuvering trajectories of land, air and ocean navigation systems. However, the SLAM assumes that the unknown environment is static and the moving objects are as noise sources.

The study of DTMO is the main part of the estimation and prediction of maneuvering trajectories in ocean navigation because moving targets are the major concern in its analysis. Even though the DTMO and SLAM are developed as two independent research directions, they can be complementary to each other in navigation systems [10].

The main functionalities of the DTMO systems are divided into three sections in recent studies [11]: Scan unit, Target Classification unit and Target Tracking & Behavior Prediction unit. The Scan unit consists of the instrumentations that are used for identification of the targets. The Target Classification unit consists of a classification of the targets with respect to the geometrical shapes and sizes. Finally, the Target Tracking & Behavior Prediction is used for estimation of the target current states and prediction of the target future states.

Identification of an accurate mathematical model for the maneuvering target is an important step in estimation and prediction of future trajectories in ocean navigation. When a single model cannot capture the required behavior of the target, the multiple model approaches is also proposed in several studies [12]. In general, maneuvering target tracking models that are used in recent literature can be divided into three categories considering the dimensional space [7]: 1D, 2D and 3D models. While 3D models are popular applications of the air and submersible navigation systems and 1D and 2D models are used in land and ocean navigation systems.

Nevertheless, a formulation of an effective estimation algorithm for maneuvering target is also an important step in prediction of the future maneuvering trajectories in ocean navigation. However, the accuracy of trajectory prediction of a target depends on the adaptive capabilities in the estimation algorithm and there are several approaches can be indentified in recent literature.

A multiple model approach, a constant velocity model and constant speed turn model, with the unscented Kalman filter for curvilinear motion for tracking of maneuvering vehicle is proposed in [13]. Further 2D Laser based obstacle motion tracking in unconstrained environments, with the Kalman Filter algorithm and predicting obstacles future motion, is presented in [14].

The target tracking methods, in combination with the Particle filter and Kalman filters using the radar information is presented in [15]. Furthermore, the Neural Kalman filter for target tracking is illustrated in the study of [16].

A people tracking system that is based on the Laser range data, a multi-hypothesis Leg-Tracker, using a Kalman filter with a constant velocity model, is proposed by [17]. The 2D Laser based obstacle motion tracking in dynamic unconstrained environments using the Kalman filter algorithm [18], and Particle Filters and Probabilistic Data

 ISBN: 978-1-61208-109-0 

Associations [19] to predict targets motions are presented in the respective studies.

## III. ESTIMATION AND PREDICTION OF OCEAN VESSEL TRAJECTORIES

The main objective in this section is to develop mathematical tools for the estimation and prediction of navigation trajectories of ocean vessels. Therefore, this section is divided into three sections [20]: Target Motion Model (TMM), Measurement Model and Associated Techniques (MAT) and Trajectory Tracking and Estimation (TTE).

### A. *Target Motion Model*

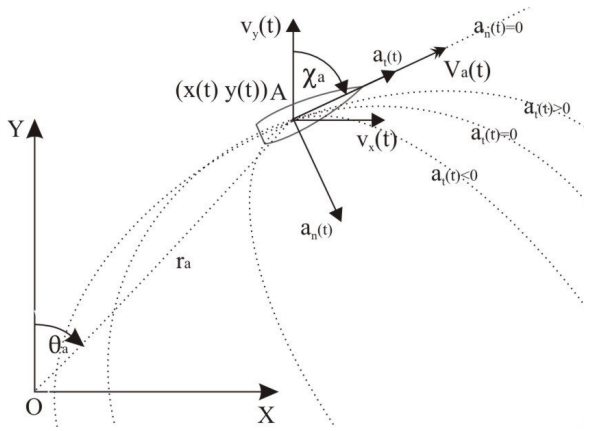



Figure 1. Curvilinear Motion Model

A suitable mathematical model for the vessel maneuvering in ocean navigation is considered in this section. The 2D kinematic model that can capture the navigation capabilities of an ocean vessel is considered during model selection process. In general, ocean vessels always follow parabolic shaped maneuvering trajectories rather than sudden motions as observed in the land and air navigation systems. Furthermore, a vessel maneuvering model is assumed to be a point target with negligible dimensions in this study. Considering the above requirements the continuous-time Curvilinear Motion Model [21] is proposed as the TMM.

The continuous-time Curvilinear Motion Model that is formulated for ocean vessel navigation is presented in Figure 1. The vessel is located in the point A. The vessel x and y positions are represented by x(t) and y(t) in continuous-time with respect to the XY coordinate system. Furthermore, the continuous-time velocity components along the x and y axis are represented by $v_x(t)$ and $v_y(t)$. The heading angle is presented by $\chi_a(t)$ and it is assumed that the vessel course and heading conditions are similar. The vessel total continuous velocity is presented by $V_a(t)$, (where $V^2_a(t) = v^2_x(t) + v^2_x(t)$) as illustrated in the figure.

On should note that there are some important features that can be observed from the Curvilinear Motion Model. As presented in the figure, when the normal acceleration $a_n(t)$ is 0 the model performs the straight line motion, when the tangential acceleration $a_t(t)$ is 0 the model performs circular motions. Furthermore $a_t(t) > 0$ and $a_t(t) < 0$ the acceleration conditions that produce parabolic navigation trajectories are also presented in the figure.

Therefore, the Curvilinear Motion Model capabilities of capturing the multi-model features are other advantages in this approach. The standard continuous-time Curvilinear Motion model can be written as:

$$\begin{aligned} \dot{\chi}_a(t) &= \frac{a_n(t)}{V_a(t)} \\ \dot{\mathbf{V}}_a(t) &= a_t(t) \\ v_x(t) &= V_a(t)\sin(\chi_a(t)) \\ v_y(t) &= V_a(t)\cos(\chi_a(t)) \end{aligned} \tag{3}$$

The summarized TMM presented on Equation (3) can be formulated as a nonlinear dynamic system model:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) + \mathbf{w}_x(t) \tag{4}$$

where

$$\mathbf{x}(t) = \begin{bmatrix} x(t) \\ v_x(t) \\ y(t) \\ v_y(t) \\ a_t(t) \\ a_n(t) \end{bmatrix}, \qquad \mathbf{f}(\mathbf{x}(t)) = \begin{bmatrix} v_x(t) \\ a_t(t)f^{vx} + a_n(t)f^{vy} \\ v_y(t) \\ a_t(t)f^{vy} - a_n(t)f^{vx} \\ 0 \\ 0 \end{bmatrix}$$

$$f^{vx} = \frac{v_x(t)}{\sqrt{v_x^2(t) + v_y^2(t)}}, \quad f^{vy} = \frac{v_y(t)}{\sqrt{v_x^2(t) + v_y^2(t)}}$$

and $\mathbf{w}_x(t)$ is the process noise that is considered as a white Gaussian distributions with 0 mean value and Q(t) covariance.

$$Q(t) = \mathrm{diag}\left[Q_x(t)\ Q_{vx}(t)\ Q_y(t)\ Q_{vy}(t)\ Q_{at}(t)\ Q_{an}(t)\right]$$

where $Q_x(t)$, $Q_{vx}(t)$ , $Q_y(t)$ , $Q_{vy}(t)$ , $Q_{at}(t)$ and $Q_{an}(t)$ are respective system state covariance values. Furthermore, the tangential $a_t(t)$ and normal $a_n(t)$ accelerations are formulated as:

$$\dot{a}_t(t) = w_{at}(t) \quad \text{with } E[a_t(t)] = a_{t0} \tag{5}$$

$$\dot{a}_n(t) = w_{nt}(t) \quad \text{with } E[a_n(t)] = a_{n0} \tag{6}$$

where $a_{t0}$ and $a_{n0}$ are mean acceleration values that are constants, and $w_{at}(t)$ and $w_{nt}(t)$ are tangential and normal

acceleration derivatives that are modeled as white Gaussian distributions with 0 mean and, $Q_{at}(t)$ and $Q_{an}(t)$ covariance values, respectively. The Jacobian of f($\mathbf{x}$(k)) can be expressed as:

$$\frac{\partial}{\partial \mathbf{x}}\left(f\left(\mathbf{x}(t)\right)\right) = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & a_t(t)f_{vx}^{vx} + a_n(t)f_{vx}^{vy} & 0 & a_t(t)f_{vy}^{vx} + a_n(t)f_{vy}^{vy} & f^{vx} & f^{vy} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & a_t(t)f_{vx}^{vy} - a_n(t)f_{vx}^{vx} & 0 & a_t(t)f_{vy}^{vy} - a_n(t)f_{vy}^{vx} & f^{vx} & -f^{vy} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (7)$$

where

$$f_{vx}^{vx} = \frac{v_x^{\,2}(t)}{\left(v_x^{\,2}(t) + v_y^{\,2}(t)\right)^{3/2}}, \quad f_{vy}^{vx} = \frac{v_y(t)v_x(t)}{\left(v_x^{\,2}(t) + v_y^{\,2}(t)\right)^{3/2}}$$

$$f_{vx}^{vy} = \frac{v_x(t)v_y(t)}{\left(v_x^{\,2}(t) + v_y^{\,2}(t)\right)^{3/2}}, \quad f_{vy}^{vy} = \frac{v_y^{\,2}(t)}{\left(v_x^{\,2}(t) + v_y^{\,2}(t)\right)^{3/2}}$$

### B. Measurement Models and Associated Technique

The measurement model is formulated as a discrete-time linear model due to availability of the ocean vessel positions usually in discrete time instants. The position values of ocean vessels can be captured by Radar or Laser based measurement systems. It is assumed that the vessel position measurement sensor is located in the position O (0,0), as presented in Figure 1. Even though the Radar or Laser based measurement systems initially capture the Polar coordinates of ocean vessels, it is assumed that the Cartesian coordinates of the position coordinates can be derived and no correlation between the position measurements. The vessel position measurements in discrete-time can be written as:

$$\mathbf{z}(k) = h(\mathbf{x}(k)) + w_z(k) \quad (8)$$

where

$$\mathbf{z}(k) = \begin{bmatrix} z_x(k) \\ z_y(k) \end{bmatrix}, \mathbf{h}(\mathbf{x}(k)) = \begin{bmatrix} x(k) & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & y(k) & 0 & 0 & 0 \end{bmatrix}$$

and $z_x(k)$ and $z_y(k)$ are measurements of x and y positions of the target vessel, and $w_y(k)$ is a white Gaussian measurement noise with zero mean and covariance R(k). The covariance R(k) can be written as:

$$R(k) = \mathrm{diag}\left[R_x(k)\ R_y(k)\right]$$

where $R_x(k)$ and $R_y(k)$ are respective measurements covariance values. The Jacobian matrix of measurement model can be written as:

$$\frac{\partial}{\partial x}\left(h\left(x(k)\right)\right) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (9)$$

### C. Trajectory Tracking and Estimation

The development of the Trajectory Tracking and Estimation (TTE) can elaborate into several directions in recent studies [20]: The single model based Kalman Filter (KF), Extended Kalman Filter (EKF), Adaptive Kalman Filter (AKF), etc.. However, the Extended Kalman Filter is proposed as an adaptive algorithm for the TTE in this study, due to the EKF capabilities of capturing the nonlinear system states of the ocean vessel navigation.

In 1960, R.E. Kalman formulated a method of minimization of a mean-least square error-filtering problem using a state space system model. The two main features of the Kalman formulation and solutions of systems are associated with, the Vector modeling of the random processes under consideration and recursive processing of the noisy measurements data [22]. However, these conditions are associated with most of the engineering problems.

The general KF algorithm is limited for application to linear systems. Therefore, the Extended Kalman Filter (EKF) is considered as the standard technique for a number of non-linear system applications. The summarized Extended Kalman Filter [23] algorithm can be written as:

#### 1) System Model

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t)) + \mathbf{w}_x(t)$$

$$\mathbf{w}_x(t) \sim N(0, Q(t)) \quad (10)$$

$$E\left[\mathbf{w}_x(t)\right] = 0,\ E\left[\mathbf{w}_x(t); \mathbf{w}_x(t)\right] = \left[Q(t)\right]$$

#### 2) Measurement Model

$$\mathbf{z}(k) = h(\mathbf{x}(k)) + w_z(k)$$

$$w_z(k) \sim N\left(0, R(k)\right),\ k = 1,2,... \quad (11)$$

$$E\left[w_z(k)\right] = 0,\ E\left[w_z(k); w_z(k)\right] = \left[R(k)\right]$$

#### 3) Error Conditions

$$\tilde{x}(k) = \hat{x}(k) - \bar{x}(k) \quad (12)$$

where $\tilde{x}(t)$ is the state error and $\hat{x}(t)$ the estimated states of the system.

*4) System Initial States*

$$\mathbf{x}(0) \sim N(\hat{\mathbf{x}}(0), P(0)) \tag{13}$$

where $\mathbf{x}(0)$ is the initial estimated values and P(0) is the initial estimated error covariance of the system states.

*5) Other Conditions*

$$E[\mathbf{w}_x(t); \mathbf{w}_z(k)] = 0 \quad \text{for all } k, t \tag{14}$$

*6) State Estimation Propagation*

$$\frac{d}{dt}\hat{\mathbf{x}}(k) = f(\hat{\mathbf{x}}(k)) \tag{15}$$

*7) Error Covariance Extrapolation*

$$\frac{d}{dt}\mathbf{P}(k) = \mathbf{F}(\hat{\mathbf{x}}(k))\mathbf{P}(k) + \mathbf{P}(k)\mathbf{F}^{T}(\hat{\mathbf{x}}(k)) + Q(k)$$
$$\mathbf{F}(\hat{\mathbf{x}}(k)) = \frac{\partial}{\partial \mathbf{x}(k)}\mathbf{F}(\hat{\mathbf{x}}(k))\Big|_{x(k)=x(k)} \tag{16}$$

where P(k) is the estimated error covariance with

$$P(k) = \text{diag}\left[P_x(k)\ P_{vx}(k)\ P_y(k)\ P_{vy}(k)\ P_{at}(k)\ P_{an}(k)\right]$$

and $P_x(k)$, $P_{vx}(k)$ , $P_y(k)$ , $P_{vy}(k)$ , $P_{at}(k)$ and $P_{an}(k)$ are respective estimated state error covariance values.

*8) State Estimate Update*

$$\hat{\mathbf{x}}(k^+) = \hat{\mathbf{x}}(k^-) + K(k)\left[\mathbf{z}(k) - \mathbf{h}_k\left(\hat{\mathbf{x}}(k^-)\right)\right] \tag{17}$$

where $\mathbf{x}(k^-)$ and $\mathbf{x}(k^+)$ are the prior and posterior estimated system states respectively, and K(k) is the Kalman gain.

*9) Error Covariance Update*

$$\mathbf{P}(k^+) = \left[1 - K(k)H_k\left(\hat{\mathbf{x}}(k^-)\right)\right]\mathbf{P}(k^-) \tag{18}$$

where $\mathbf{P}(k^-)$ and $\mathbf{P}(k^+)$ are the prior and posterior error covariance of the system state respectively.

*10) Kalman Filter Gain*

$$\mathbf{K}(k) = \mathbf{P}(k^-)H\left(\hat{\mathbf{x}}(k^-)\right)\left[H\left(\hat{\mathbf{x}}(k^-)\right)\mathbf{P}(k^-)H\left(\hat{\mathbf{x}}(k^-)\right)^{T} - R(k)\right]^{-1} \tag{19}$$

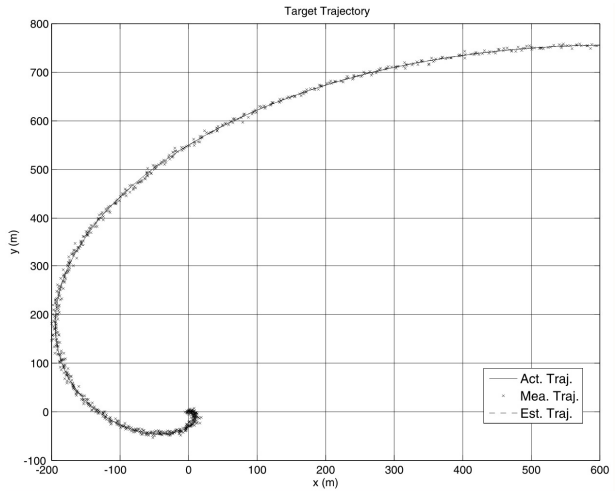



Figure 2. EKF Trajectory Estimation

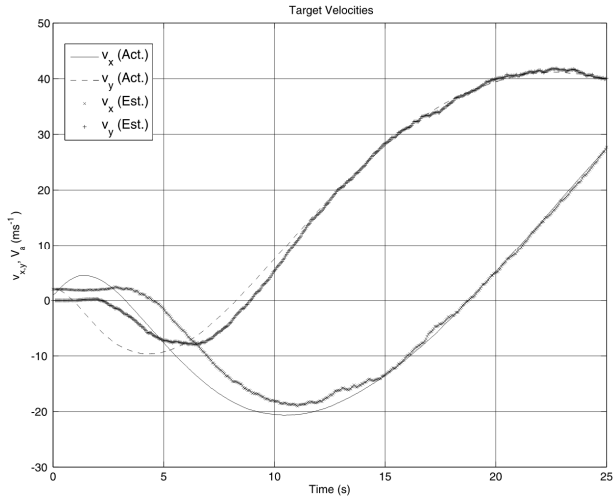



Figure 3. EKF Velocity Estimation

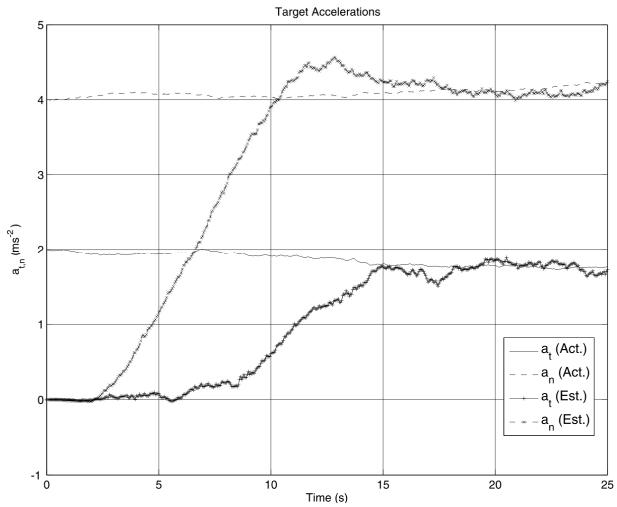



Figure 4. EKF Acceleration Estimation

## IV. COMPUTATIONAL SIMULATIONS

This section contains a detail description of the software architecture and initial state values that are considered for the simulations. The proposed EKF algorithm is tested on the MATLAB software platform and simulations are presented in Figures of 2, 3 and 4.

The values that are considered in the computational simulations of the EKF simulations can be presented as: The actual start position x (0) = 0 (m) and y (0) = 0 (m) of the ocean vessel is considered. Then the initial velocity components of $v_x$ (0) = 1 ($ms^{-1}$) and $v_y$ (0) = 2 ($ms^{-1}$) are assigned and the actual mean accelerations $a_{t0} = 2(ms^{-2})$ and $a_{n0} = 4(ms^{-2})$ are assumed. The initial estimated position as $\hat{x}(0) = 3$(m) and $\hat{y}(0) = 0$(m) are considered. The estimated initial velocity components of $\hat{v}_x(0) = 2(ms^{-1})$ and $\hat{v}_y(0)$ = 0 ($ms^{-1}$) values are considered for the EKF algorithm. Furthermore, the initial estimated acceleration components of $\hat{a}_t(0) = 0(ms^{-2})$ and $\hat{a}_n(0) = 0(ms^{-2})$ are considered. The sampling time used in the EKF estimation is 0.01 (s).

The system state covariance values are assigned as $Q_x(t) = Q_{vx}(t) = Q_y(t) = Q_{vy}(t) = 0.1$ and $Q_{at}(t) = Q_{an}(t) = 0.01$, with the assumptions of position, velocity and acceleration component covariance values are uncorrelated. Similarly the initial estimated error covariance values are assigned as $P_x(0) = P_{vx}(0) = P_y(0) = P_{vy}(0) = P_{at}(0) = P_{an}(0) = 0.01$, with assumptions of position, velocity and acceleration estimation error covariance components are uncorrelated. The covariance values for the measurements are assigned as $R_x(t) = R_y(t) = 10$ with assumption of position measurement covariance components are uncorrelated.

The computational simulations of the trajectory estimations for a maneuvering target vessel using the EKF algorithm are presented in Figure 2. The figure represents the actual trajectory (Act. Traj), Measured trajectory (Mea. Traj.) and Estimated trajectory (Est. Traj.) of the ocean navigation. As noted from the figure, the EKF estimates the vessel maneuvering trajectory successfully. The vessel velocity components of $v_x(t)$ and $v_y(t)$ of actual and estimated are presented in Figure 3. Furthermore, the figure represents the Actual (Act.) and Estimated (Est.) velocities for each velocity components. The successful velocity estimation values are also achieved by the EKF algorithm as presented in the figure within 15 (s) of time interval.

The Estimated (Est.) accelerations of $a_t(t)$ and $a_n(t)$ values are presented in Figure 4 with respect to the Actual (Act.) acceleration values. Furthermore, the figure represents the convergence of the Estimated accelerations into the Actual accelerations for normal and tangential acceleration components within 15 (s).

## V. CONCLUSION

The satisfactory prediction of ocean vessel positions, velocities and accelerations are achieved by the EKF estimation that is working as an adaptive filter incorporated with the Curvilinear Motion Model and linear measurement model. As presented in Figure 4, the convergence of the estimated accelerations into actual accelerations within approximate time interval of 15(s). Therefore, the estimated velocities and acceleration components can be used for the future maneuvering trajectory prediction of ocean vessel navigation.

The estimated values of the velocity components have small variations around the actual values and that affect on the acceleration estimations. Hence, smoothing techniques can be used for better convergence of the estimated values of system states into the actual values. The improved system states can be used for better prediction of ocean vessel navigation trajectories within smaller time intervals.

Furthermore, it is assumed that the mean acceleration components are constant values with a white Gaussian noise in this study. However, this assumption may not always be realistic and changing acceleration conditions can be observed in ocean navigation. Even when real ocean vessel navigation consists of changing acceleration conditions, the formulations presented in this paper can still hold with the assumptions of constant acceleration within short time intervals.

One should note that the velocity and acceleration estimation values are achieved by only the noisy position measurements that are collected from the vessel navigation, which is the main contribution in this approach. However, the improved formation of the EKF algorithm, with the smoothing techniques for the fast convergence into actual accelerations, is proposed as the further developments in this study.

## ACKNOWLEDGMENT

This work has been made within the project "Methodology for ships maneuverability tests with self-propelled models", which is being funded by the Portuguese Foundation for Science and Technology (Fundação para a Ciência e Tecnologia) under contract PTDC/TRA/74332 /2006. The research work of the first author has been supported by a Doctoral Fellowship of the Portuguese Foundation for Science and Technology (Fundação para a Ciência e Tecnologia) under contract SFRH/BD/46270/2008.

## REFERENCES

[1] EMSA Report, "European maritime safety agency," http://www.emsa.europa.eu/, [retrieved: August, 2010].

[2] Directive 2002/59/EC of the European Parliament and of the Council, "Establish a community vessel traffic monitoring and information system and repealing council directive 93/75/eec," *Official Journal of the European Communities*, no. 208, pp. 10–27, 2002.

[3] F. Farina, "Introduction to radar signal & data processing : The opportunity," RTO SET Lecture Series on : Knowledge-Based Radar Signal and Data Processing, November 2004, RTO-EN-SET-063, Gdansk, Porland.

[4] L. P. Perera and C. Guedes Soares, "Laser measurement system based maneuvering target tracking formulated by adaptive competitive neural networks," in *In Proc. 2nd ADPTIVE*, Lisbon, Portugal, 2010, (In Print).

[5] K. Hasegawa, "Advanced marine traffic automation and management system for congested waterways and coastal areas," in *Proceedings of International Conference in Ocean Engineering (ICOE)*, 2009, pp. 1–10.

 ISBN: 978-1-61208-109-0 

[6] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," in *Proceedings of the IEEE*, vol. 85, no. 1, 1997, pp. 6-23.

[7] X. R. Li and V. P. Jilkov, "Survey of maneuvering target tracking - Part I : Dynamic models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 39, no. 4, pp. 1333–1364, 2003.

[8] L. P. Perera, J. P. Carvalho, and C. Guedes Soares, "Fuzzy-logic based parallel collisions avoidance decision formulation for an ocean navigational system," In *Proc. 8th IFAC Conference on Control Applications in Marine Systems*, Rostock, Germany, 2010, (In Print).

[9] L. P. Perera, J. P. Carvalho, and C. Guedes Soares, "Bayesian Network based sequential collision avoidance action execution for an ocean navigational system," In *Proc. 8th IFAC Conference on Control Applications in Marine Systems*, Rostock, Germany, 2010, (In Print).

[10] C. Wang and C. Thorpe, "Simultaneous localization with detection and tracking of moving objects," in *IEEE int. Conf. on Robotics and Automation*, Washington DC, 2002, pp. 2918–2924.

[11] A. Mendes, L. C. Bento, and U. Nunes, "Multi-target detection and tracking with a laserscanner," in *2004 IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004, pp. 796–800.

[12] M. Athans and C. B. Chang, "Adaptive estimation and parameter identification using multiple model estimation algorithm," Technical Note 1976-28, MIT Lincoln Lab., Lexington, Mass., 1976.

[13] Y. Kim and K. Hong, "An IMM algorithm for tracking maneuvering vehicles in an adaptive cruise control environment," *International Journal of Control, Automation and Systems*, vol. 2, no. 3, pp. 310–318, 2004.

[14] M. Berker, R. Hall, S. Kolski, K. Macek, R. Siegwart, and B. Jensen, "2d laser-based probabilistic motion tracking in urban-like environments," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 31, no. 2, pp. 83–96, 2009.

[15] D. Angelova and L. Mihaylova, "Joint target tracking and classification with particle filtering and mixture kalman filtering using kinematic radar information," *Digital Signal Processing*, vol. 16, pp. 180–204, 2006.

[16] S. C. Stubberud and K. A. Kramer, "Kinematic prediction for intercept using a neural kalman filter," in *Proceedings of the 16th IFAC World Congress*, Prague,Czech Republic, 2005.

[17] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities," in *IEEE International Conference on Robotics and Automations*, CA, USA, 2008, pp. 1710–1715.

[18] M. Berker, R. Hall, S. Kolski, K. Macek, R. Siegwart, and B. Jensen, "2D laser-based probabilistic motion tracking in urban-like environments," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 31, no. 2, pp. 83–96, 2009.

[19] A. Almeida, J. Almeida, and R. Araujo, "Real-time tracking of multiple moving objects using particle filters and probabilistic data association," *Automatika*, vol. 46, no. 1–2, pp. 39–48, 2005.

[20] X. R. Li and V. P. Jilkov, "A survey of maneuvering target tracking - Part IV : Decision-based methods," in *Proceedings of SPIE Conference on Signal and Data Processing of Small Targets*, Orlando, FL, USA, 2002, pp. 4728–4760.

[21] R. A. Best and J. P. Norton, "A new model and efficient tracker fro target with curvilinear motion," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 33, no. 3, pp. 1030–1037, July 1997.

[22] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals andApplied Kalman Filtering*. New York, USA: John Wiely & Sons, 1997.

[23] A. Gelb, J. F. Kasper, Jr., R. A. Nash, Jr., C. F. Price, and A. A. Sutherland, Jr., *Applied Optimal Estimation*. MA. USA: The MIT Press, 2000.

 ISBN: 978-1-61208-109-0 

# Adaptive Cooperative Multi-Hop Transmission in Ad Hoc Networks

Wasimon Panichpattanakul, Beatrice Paillassa, Benoit Escrig
University of Toulouse, IRIT Laboratory – ENSEEIHT
Toulouse, FRANCE
wasimon.panichpattanakul, beatrice.paillassa, benoit.escrig@enseeiht.fr

Daniel Roviras
Conservatoire National des Arts et Métiers
Paris, FRANCE
daniel.roviras@cnam.fr

***Abstract*— Cooperative communication techniques have been proposed in order to improve the quality of the received signals at the receivers by using the diversity added by duplication of signals sent by relay terminals situating between each transmission pair. This paper proposes an adaptive cooperation technique for frame transmissions in Ad Hoc networks that is compatible to both of the basic access mode and the optional access mode of IEEE 802.11 Medium Access Control (MAC) protocol. The transmission mode for each data frame is adaptively switched between a cooperative mode and a non-cooperative mode based on the absence of acknowledge (ACK) frame. Simulations show that transmission performance is improved by decreasing the number of re-transmissions due to frame errors; thus, chances of multi-hop mode transitions that are costly in time and bandwidth are alleviated. The analysis of the proposition performance indicates the interest of the adaptation paradigm. It puts forward that, in addition to the channel quality parameter, the channel availability parameter must be concerned in the adaptation process.**

***Keywords-adaptive cooperation; cooperative transmissions; IEEE 802.11***

## I. INTRODUCTION

In wireless communications, fading causes errors on data transmissions. Based on IEEE 802.11 MAC standard, re-transmission processes are required when error data frames are detected. Obviously, re-transmissions increase delay and decrease the packet delivery ratio (PDR) of the networks. More precisely, in multi-hop networks, if the re-transmission counter (Re-Tx) reaches the threshold, a route recovery process is activated. For example, considering Ad Hoc On-Demand Distance Vector (AODV) routing protocol [1], the route recovery process is done by an AODV source-initiated route re-discovery method. The source terminal (S) broadcasts a route request (RREQ) packet to re-find a route to the destination terminal (D). The RREQ packet will be re-broadcasted through the network. Therefore, the network is flooded and is led to network congestion problems.

In addition, if the route re-discovery process occurs when the direct path (S-to-D) is dropped, instead of receiving the RREQ packet from S, D receives the RREQ packet from an intermediate terminal (I) locating between S and D terminals. Thus, the transmission mode is switched from direct transmission mode to multi-hop transmission mode as shown in Fig.1. Rather than directly transmits a data frame from S to D in one time slot, the multi-hop transmission requires two time slots to send this data frame from S to I and from I to D, respectively. Therefore, similar to re-transmissions, multi-hop transmissions also increase the delay and decrease the PDR of the networks.

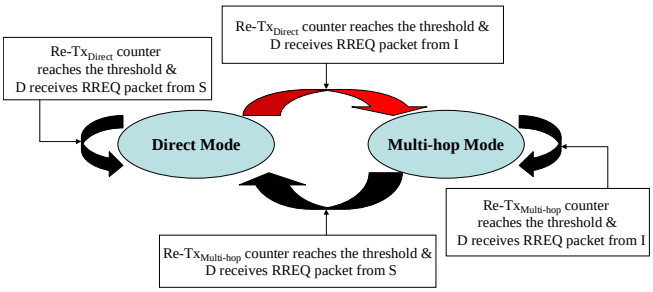



Figure 1. Transmission mode transitions.

Since multi-hop mode transitions happen when the Re-Tx counter of the direct mode transmission (Rx-Tx$_{Direct}$) reaches the threshold, transmission performance of the direct mode must be improved in order to reduce the number of re-transmissions. Multiple-input Multiple-output (MIMO) is an example of transmission techniques that have been proposed to improve transmission performance in wireless communications. MIMO provide advantages of spatial diversity by uncorrelated signal components generated from antenna array at a source terminal and/or a destination terminal. However, each antenna in the antenna array must be separated at least $\lambda/2$ in order to provide independent signals. $\lambda$ is the wave length of the system signal and it can be calculated as follows;

$$\lambda = \frac{c}{f_c} \quad (1)$$

Where c is a speed of light ($3x10^8$ m/s) and $f_c$ is a carrier frequency. Thus, for commonly used 2.4GHz frequency band, the space between antennas at 6.125 cm is required. These requirements make MIMO technique to be impractical to employ in networks with small wireless terminals such as sensor networks. In addition, MIMO requires multiple antennas, which are costly. For these kinds of contexts, cooperative communications provide an interesting

 ISBN: 978-1-61208-109-0 

alternative that can gain benefits of spatial diversity while a single antenna is required on each terminal. Illustrations of MIMO and cooperative transmissions are respectively shown in Fig. 2a and Fig. 2b.

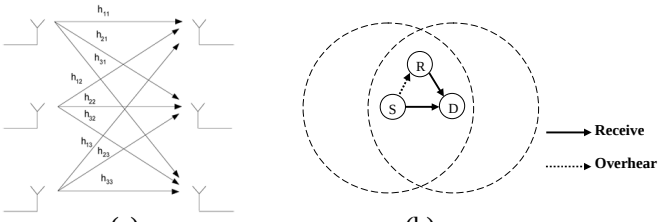



Figure 2. Illustrations of (a) MIMO and (b) Cooperative transmissions.

Cooperative transmissions have been introduced by [2] to [4]. The concept of cooperative transmissions is to exploit the broadcast nature of the wireless medium and to transform single-antenna neighbour nodes, to work as virtual antenna arrays. As shown in Fig. 3, cooperative transmissions (Fig. 3b) utilize more medium than non-cooperative transmissions (Fig. 3a) when the channel quality of the direct path (S-to-D) is good. However, if the channel quality of the direct path is dropped, non-cooperative transmissions with re-transmission processes (Fig. 3c) consumes more medium than cooperative transmissions. Therefore, cooperative transmissions are interesting and should be used when the channel quality of the direct path is dropped. Rather than remain the transmission mode in cooperative mode (named fixed cooperative transmission), cooperative transmissions should be able to switch their transmission modes between cooperative mode and non-cooperative one. These cooperative transmissions are called adaptive cooperative transmissions.

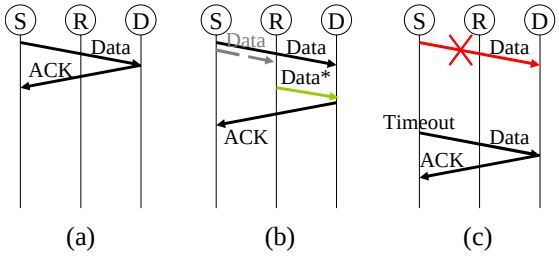



Figure 3. Message flows of (a) Non-cooperative transmissions (b) Cooperative transmissions and (c) Non-cooperative transmissions with re-transmission processes.

In adaptive cooperative transmissions, generally inspired from the IEEE 802.11 MAC standard [5], the activation and deactivation of these cooperative transmission modes require extra control frames, which are modified from Request-To-Send (RTS) or Clear-To-Send (CTS) frames [6] and [7] and/or are created in new frame formats [8] and [9]. These adaptive cooperative transmissions cannot be implemented in IEEE 802.11 networks with basic access mode and also have interoperability problems with legacy systems.

For example, the message flow of an adaptive cooperative transmission protocol called CoopMAC [6] is shown in Fig. 4. CoopMAC uses CoopRTS frames (modified from RTS frames), HTS frames (Helper ready To Send, modified from CTS frame), and CTR frames to activate and deactivate cooperative transmission modes among terminals. Therefore, CoopMAC can be implemented only in the optional access mode of IEEE 802.11 networks.

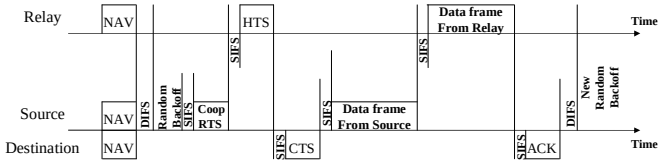



Figure 4. Message flows in CoopMAC protocol [6].

To overcome these problems, we propose a simple but effective transmission method called Adaptive Cooperation Multi-Hop Transmission (ACMHT). In order to have our proposition compatibly work with terminals without cooperative functionality in legacy systems, only some process modifications are required and affect only the nodes with cooperative functionality. In addition, our method does not use or modify RTS or CTS frames; therefore, it can work compatibly with both of the basic and the optional access methods of the IEEE 802.11 MAC protocol. Transmission mode of ACMHT can be switched between cooperative and non-cooperative mode based on the absence of ACK frames.

Our proposition intends to increase the link quality by a cooperative mechanism, and also to prevent unnecessary routing processes such as route maintenance and re-route discovery. In addition, the proposed method alleviates probability of multi-hop mode transitions in order to reduce costs of multi-hop mode transmissions. To evaluate the interest of our proposition, the transmission performance in term of PDR is considered.

The rest of the paper is organized as follows. In Section II, details of the proposition are presented. Section III is devoted to the system model while Section IV concerns with simulation results and analysis. Finally, the conclusion is drawn in Section V.

## II. ADAPTIVE COOPERATIVE MULTI-HOP TRANSMISSION

Our proposition is designed for a WiFi network using an IEEE 802.11 MAC protocol. For interoperability purposes, rather than specifying a new protocol, we decided to derive benefit from the handshaking access mechanisms to activate or deactivate the cooperative mode. Mechanisms of ACMHT when it works with the basic access method (also called two-way handshaking; Data/ACK) are shown in Fig. 5a and Fig. 5b and with the optional access method (also called four-way handshaking; RTS/CTS/Data/ACK) are shown in Fig. 5c and Fig. 5d. S, R, and D stand for Source, Relay, and Destination respectively. R is assumed to be chosen and is located in the transmission ranges of S and D. Fig. 5a and Fig. 5c represent ACMHT message flows when it works in a non-cooperative transmission mode and when it works in a cooperative transmission mode are shown in Fig. 5b and Fig. 5d.

The proposition is adaptive because its transmission mode is able to switch between a direct mode and a cooperative multi-hop mode. The appearance of an ACK

frame informs R that the direct transmission is successful; thus, cooperative multi-hop mode is automatically turned off. The network transmission mode rests at the direct transmission mode. R remains quiet and S continues to transmit its next data frame in the direct path.

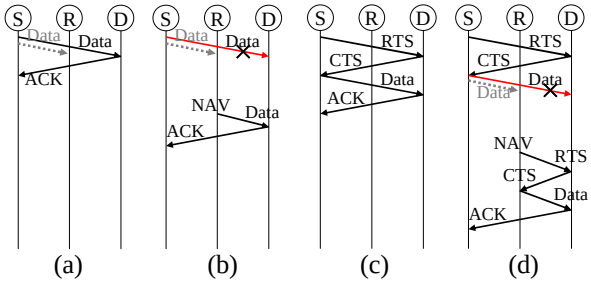



Figure 5. Message flows of our proposition (ACMHT).

On the contrary, in Fig. 5b and 4d, when D fails to decode a data frame and the network allocation vector (NAV) of R reaches to zero, the cooperative multi-hop mode transmission of the ACMHT is automatically turned on. The transmission mode of the network is temporally switched from direct mode to cooperative multi-hop mode as shown in Fig. 6. Without any changes in the header, R helps S to forward the data to D, and then the transmission mode of the network is automatically switched back to the direct mode. If D successfully decodes the data sent from R, it replies an ACK back to S. The Re-Tx counter at S is reset, and then S sends its next data frame. When the Re-Tx counter is reset, chances of multi-hop mode transition are alleviated.

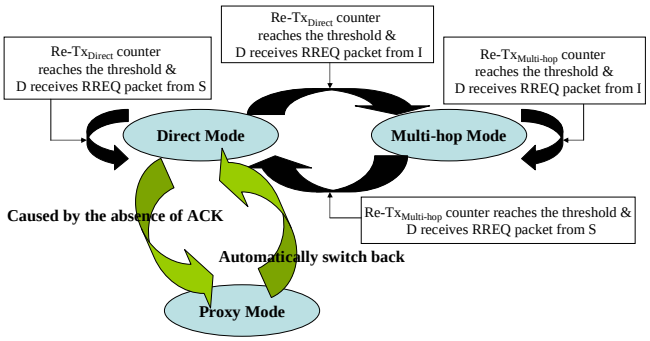



Figure 6. Transmission mode transition of ACMHT.

A MAC layer table is specified at terminal R in order to allow R to be able to filter and relay data frames sent from S to D correctly. MAC addresses of the transmission pair (S and D) are indicated in the table. These addresses are acquired by upper layer protocols such as Hello or routing protocols in the network layer. For data relaying, MAC layer relaying is chosen instead of network layer forwarding. R acts as a dynamical bridge since forwarded data frames do not need to be sent up to the network layer; thus, queuing delays and processing delays are alleviated. R directly forward exactly the same data frame (received from S) to D through its MAC layer. In addition, after forwarding the data, R does not need to wait ACK frames from D.

If adaptive cooperative transmission done by R is also fail or the ACK frame sent from D is lost, S waits until its retransmission timer reaches to zero, then the data is retransmitted. To prevent collision between re-transmissions done by S and cooperative multi-hip transmissions done by R, S must extend its timeout at least two times of the value indicated in IEEE 802.11 MAC standard.

For simplicity, similar to [6] [7] [10] and [11], in our proposition, the received signals at the destination terminal transmitted by S and R are not combined. If signal combinations in signal-level are needed, signal combiners such as maximum ratio combiners require fading amplitudes and phase compensations of the source and the relay terminals at both of transmitter and receiver sides [12]. These requirements cause system complexities. Moreover, additional hardware such as a signal combiner at the receiver side is required and it gains cost to the system.

## III. System Model

The performance evaluation of the proposed method is done by simulations and compared with a non-cooperative transmission. NS 2.30 simulator is used [13]. Effects of channel quality and channel availability to ACMHT performance are studied. Three scenarios of 5-terminal networks (see Fig. 7) and a scenario of a 9-terminal network (see Fig. 8) are simulated. In Fig. 7a, a scenario in which only the relay terminal (R) is interfered by an A-B transmission pair is presented. Assume that the channel between A and B is perfect. Scenarios that all terminals (S, R, and D) are interfered and only R is not interfered are illustrated in Fig. 7b and Fig. 7c respectively. Note that terminals locating in the interference area cannot correctly decode received signals but they are interfered.

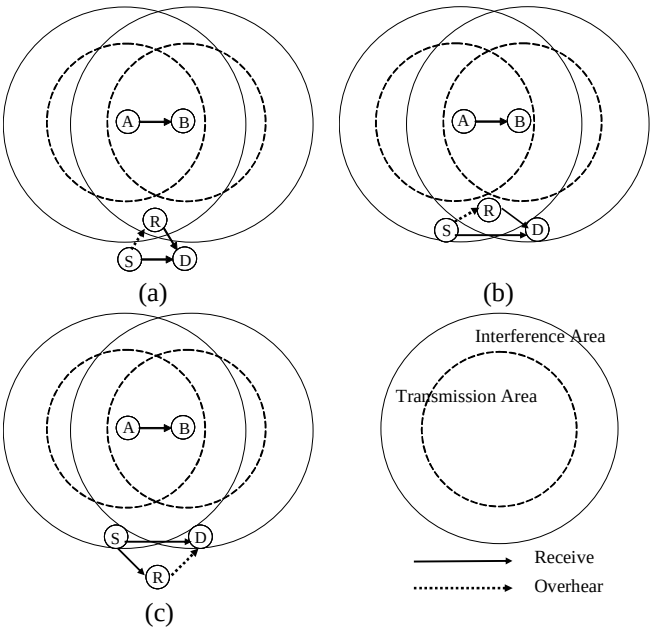



Figure 7. Three scenarios of 5-terminal networks

In Fig. 8, every case presented in Fig. 7 is included. There are three transmission pairs; i.e. S1 to D1, S2 to D2, and S3 to D3 with one relay terminal (i.e., R1, R2, and R3) for each transmission pair.

 ISBN: 978-1-61208-109-0 

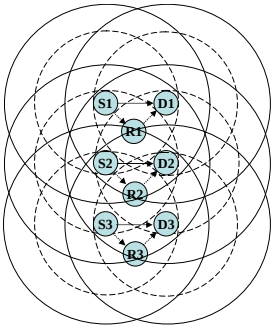



Figure 8. A scenario of a 9-terminal network.

To study the effects of channel quality to the ACMHT transmission performance in term of PDR, channel quality in term of error probabilities in the direct path (Si to Di) and the multi-hop paths (Si to Ri and Ri to Di) are varied. The frame error probabilities of the direct path (P1) are set at 0.1 and 0.2 per frame, while those of the multi-hop paths (P2) varied from 0.025 to 0.4 per frame. For physical channels, the two ray ground propagation model is used while IEEE 802.11 [5], and AODV [1] are used as the MAC, and the routing protocols. The User Datagram Protocol (UDP) agents are created to send Constant Bit Rate (CBR) traffic with data rate 448kbps and packet size equals to 210 bytes. The simulation time is 300 seconds.

## IV. Simulation Results and Analysis

In the first scenario of the 5-terminal networks, where only the terminal R is interfered by the A-B transmission pair, there is no transmission state transition in both of non-cooperative and ACMHT transmissions; thus, the percentages of data frames sent in multi-hop mode equal to zero as shown in Fig. 9. The x-axis represents values of P1 over P2 (P1/P2). The frame error probability of the direct path (P1) is set at 0.1 and 0.2 per frame and the frame error probability of each proxy path (P2) is varied from 0.03 to 0.4 per frame.

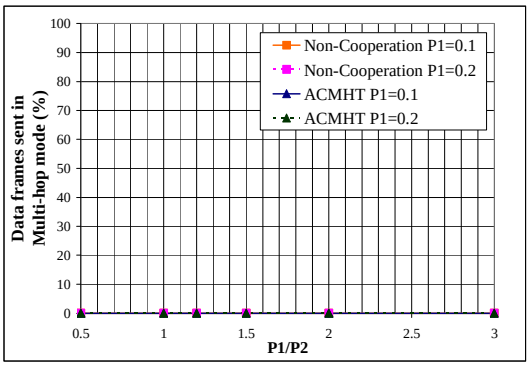



Figure 9. The percentage of data frames sent in multi-hop mode in scenario 1 of the 5-terminal networks.

PDRs of the systems with non-cooperative and ACMHT transmission in different link quality configurations are shown in Fig. 10. The PDRs of ACMHT are less than those of the non-cooperative transmission because of the effect of the extended timeout in ACMHT. If the quality of the multi-hop paths is not good and R cannot relay data to D efficiently, S in ACMHT has to re-transmit the data with the extended timeout, which causes longer delay comparing to the re-transmission processes in non-cooperative transmissions. Therefore, in non-cooperative transmissions, if there is no transmission state transition from direct mode to multi-hop mode, ACMHT is not interesting. The PDRs of the non-cooperative transmission are nearly constant because all data are sent in the direct mode; thus, the performance of the system is only function of the link quality of the direct path. The increasing of P2 does not affect the performance.

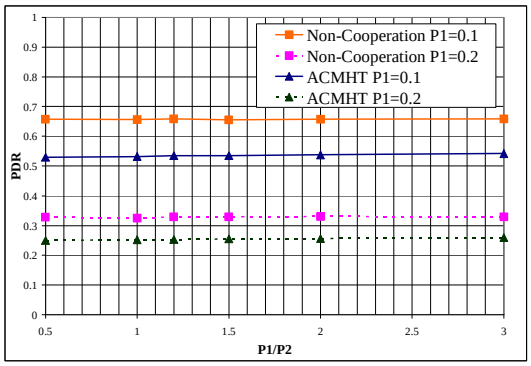



Figure 10. PDR of non-cooperative and ACMHT transmissions in scenario 1 of the 5-terminal networks.

In the second scenario of the 5-terminal networks, where all terminals are interfered by the A-B transmission pair, transmission state transitions occur in both of non-cooperative and ACMHT transmissions (see Fig. 11). However, the percentage of data frames in ACMHT that are sent in multi-hop mode is very less compared with the non-cooperative transmission. The result confirms that ACMHT can alleviate multi-hop transmission mode transitions.

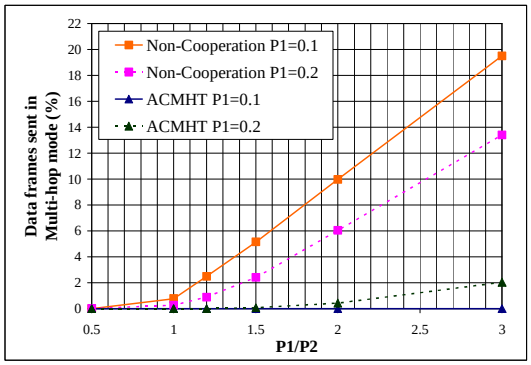



Figure 11. The percentage of data frames sent in multi-hop mode in scenario 2 of the 5-terminal networks.

In Fig. 12, ACMHT generally has higher PDRs than those of the non-cooperative transmission. Thus, we can conclude that ACMHT is interesting when every terminal has same condition of channel availability and there are chances of transmission mode transition in non-cooperative transmission to transit from direct mode to multi-hop mode.

On the left-hand side of Fig. 12 when P1= 0.2, the PDR of ACMHT is lower than that of non-cooperative

transmission because channel qualities of the multi-hop paths are very poor. This problem leads to two major drawbacks. First, when R missed-hears ACK packets, it competes with S to transmit data; thus, the collisions are occurred. Second, R is not able to help S on data relaying because it is unable to decode the data frame sent from S; therefore, D has to wait for the re-transmission done by S after the extended timeout, which is twice longer than the non-cooperative technique.

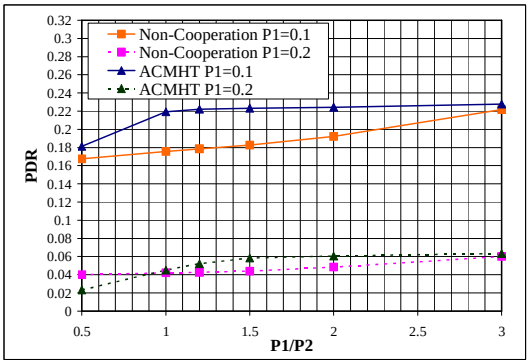



Figure 12. PDR of non-cooperative and ACMHT transmissions in scenario 2 of the 5-terminal networks.

In the third scenario of the 5-terminal networks, where only the terminal R is not interfered by the A-B transmission pair, transmission state transitions occur in non-cooperative transmission but not in ACMHT. The percentage of data frames in ACMHT that are sent in multi-hop mode equals to zero while non-cooperative transmission has high percentage of data frames sent in multi-hop mode, as shown in Fig. 13. Thus, probabilities of multi-hop transmission mode transitions are alleviated.

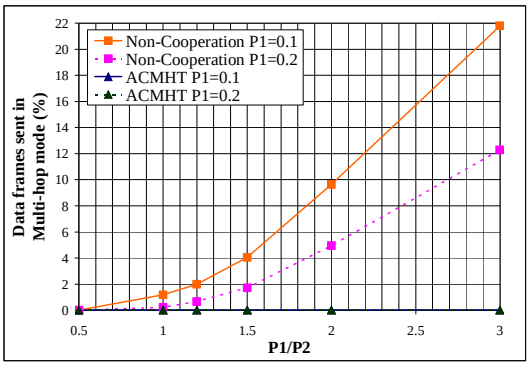



Figure 13. The percentage of data frames sent in multi-hop mode in scenario 3 of the 5-terminal networks.

Similar to the second case, ACMHT is interesting when there are chances of transmission state transition from direct mode to multi-hop mode. In addition, since R is not interfered by the A-B transmission pair, it can well perform on data relaying. Therefore, ACMHT is outperformed and has higher PDR compared to the non-cooperative transmission as shown in Fig. 14 when P1= 0.1. However, on the right-hand side of Fig. 14 when P1= 0.2 and multi-hop paths have very high channel qualities, ACMHT yields lower PDR than that of the non-cooperative transmission because the terminal D in ACMHT has to reply ACK frames through the direct path with P1= 0.2 while the non-cooperative transmission works in multi-hop mode and its ACK frames are sent through the multi-hop paths with P2 < 0.1.

In the third scenario of the 5-terminal networks, we can conclude that ACMHT is interesting when terminal R has good condition on channel availability and there are chances of transmission mode transition in non-cooperative transmission to transit from direct mode to multi-hop mode. However, if the channel qualities of the multi-hop paths are much higher than that of the direct path, ACMHT should switch its transmission mode from direct mode and cooperative multi-hop mode to multi-hop mode.

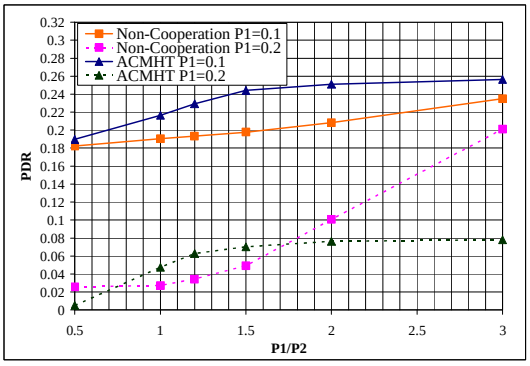



Figure 14. PDR of non-cooperative and ACMHT transmissions in scenario 3 of the 5-terminal networks.

In the 9-terminal network, the x-axis represents values of P1 over P2 (P1/P2). P1 is set at 0.1 and 0.2 per frame and P2 is varied from 0.025 to 0.4 per frame. Transmission state transitions occur in both of non-cooperative and ACMHT transmissions. However, the percentage of data frames in ACMHT that are sent in multi-hop mode is very less compared with non-cooperative transmission (see Fig. 15). Thus, the probabilities of multi-hop transmission mode transitions are also alleviated in the 9-terminal network.

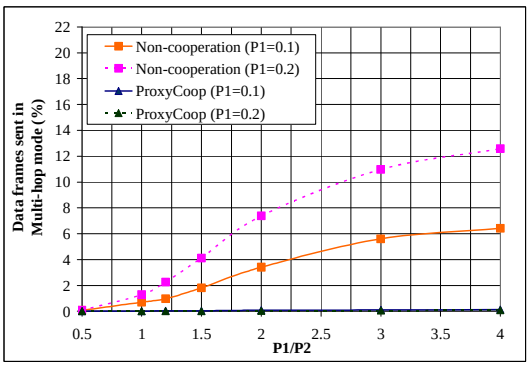



Figure 15. The percentage of data frames sent in multi-hop mode in the 9-terminal network.

In Fig. 16, on the left-hand side, when the link qualities of the proxy paths are worse than those of the direct paths, the PDR of ACMHT is lower than those of the non-cooperative transmissions because of two major reasons. First, because of the collisions generated by R when it

missed-hears ACK packets. Ri competes with Si on data transmissions. Second, due to the extended re-transmission time introduced by the inefficient relay transmission; i.e., Ri should be activated to help Si, but it is also unable to decode the data frame; thus, Di has to wait for the re-transmission done by Si after the extended timeout, which is twice longer than that of the non-cooperative technique, reaches to zero. Nevertheless, when the link qualities of the proxy paths are increased, the PDRs of ACMHT are continually increased. In some ranges of P1/P2, the ACMHT provides higher PDRs than those of the non-cooperative transmissions.

In contrast, when the link qualities of the proxy paths are increased, the PDRs of for non-cooperative transmissions are decreased due to multi-hop transmission delays. However, when multi-hop paths have very high channel qualities compared to the direct path, transmissions through multi-hop paths are more interesting than re-transmissions through the direct paths with low channel qualities. Thus, on the right-hand side of Fig. 16, the PDRs of non-cooperative transmissions are increased when the number of multi-hop transmissions is increased.

Similar to the third scenario of the 5-terminal networks, the crossing point on the right-hand side of Fig. 16 when P1= 0.2 and multi-hop paths have very high channel qualities, ACMHT yields a little bit lower PDR than that of the non-cooperative transmission because the terminal Di in ACMHT has to reply ACK frames through the direct path with P1= 0.2 while the non-cooperative transmission works in multi-hop mode and its ACK frames are sent through the multi-hop paths with P2 < 0.053.

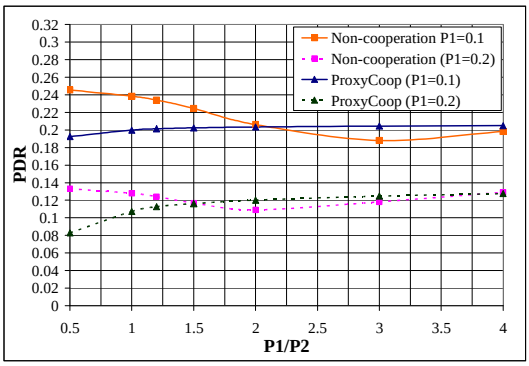



Figure 16. PDR of non-cooperative and ACMHT transmissions in the 9-terminal network.

Therefore, to improve the performance of ACMHT, rather than to switch the transmission mode of ACMHT based only on the absence of ACK frames, both of channel qualities and channel availabilities of the direct path and multi-hop paths must be considered.

## V. CONCLUSION

In this paper, an adaptive cooperative multi-hop transmission that can work compatibly with the legacy systems and is compatible to both of the basic and the optional access methods of the IEEE 802.11 MAC protocol is proposed. Beyond the proposition, the interest of the presented work concerns the study of the proposition validity that leads to determine some adaptation rules.

From simulation results, it is shown that ACMHT transmission mode must be adaptable. The proposed method outperforms the non-cooperative transmissions in terms of transmission performance (evaluated by PDR), when channel distributions of the direct path (S-to-D) can cause multi-hop mode transitions in non-cooperative transmissions and a good relay is selected. A good relay means a relay terminal having high channel availability and high channel quality of its cooperative multi-hop paths (S-to-R and R-to-D)

Thus, the control protocol in charge of relay selection (AODV routing protocol for Ad Hoc networks or Hybrid Wireless Mesh Protocol (HWMP) for wireless mesh networks for examples) has to collect information on channel qualities by measuring the frame SNR, and we also conclude that it has to collect information of channel availabilities by measuring for example the number of frames overheard by each potential relay terminal.

## REFERENCES

[1] C. Perkins, E. B. Royer, and S. Das. “Ad hoc On-Demand Distance. Vector (AODV) Routing,” *RFC 3561, IETF Network Working Group*, July 2003.

[2] A. Sendonaris, E. Erkip, and B. Aazhang, “User cooperation diversity-Part I: System description,” *IEEE Trans. Commun.*, vol. 51, no. 11, pp. 1927-1938, Nov. 2003.

[3] A. Sendonaris, E. Erkip, and B. Aazhang, “User cooperation diversity-Part II: Implementation aspects and performance analysis,” *IEEE Trans. Commun.*, vol. 51, no. 11, pp. 1939-1948, Nov. 2003.

[4] J. N. Laneman, D. N. C. Tse, and G. W. Wornell, “Cooperative diversity in wireless networks: Efficient protocols and outage behavior,” *IEEE Trans. Inform. Theory*, vol. 50, pp. 3062-3080, Dec. 2004.

[5] IEEE Standard for Information Technology-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Issue, June 2007, <http://ieeexplore.ieee.org/servlet/opac?punumber=4248376> 25.08.2010.

[6] P. Liu, Z. Tao, S. Narayanan, T. Korakis, and S. S. Panwar, “CoopMAC: A Cooperative MAC for Wireless LANs,” *IEEE Trans. Select. Areas of Commun.*, vol. 25, pp. 340 - 354, Feb. 2007.

[7] H. Zhu, G. Cao, “rDCF: A Relay-Enabled Medium Access Control Protocol for Wireless Ad Hoc Networks,” *IEEE Trans. Mobile Computing*, vol. 5, no. 9, pp. 1201-1214, July 2006.

[8] A. Azgin, Y. Altunbasak, G. AlRegib, “Cooperative MAC and Routing Protocols for Wireless Ad Hoc Networks”, *in Proc. IEEE GLOBECOM '05*, Dec. 2005, vol. 5, pp. 2854-2859.

[9] J. Zhang, Q. Zhang, and W. Jia, “VC-MAC: A cooperative MAC protocol in Vehicular Networks,” *IEEE Trans.Veh. Tech.*, vol. 58, no. 3, pp. 1561 – 1571, March 2009.

[10] P. Liu, Z. Tao, and S. Panwar, “A cooperative MAC protocol for wireless local area networks,” *in Proc. IEEE ICC '05*, May 2005, vol. 5, pp. 2962 – 2968.

[11] T. Korakis, S. Narayanan, A. Bagri, and S. Panwar, “Implementing a Cooperative MAC Protocol for Wireless LANs,” *in Proc. IEEE ICC '06*, June 2006, vol. 10, pp. 4805 – 4810.

[12] J. G. Proakis, Digital Communications, 3rd ed. New York: McGraw-Hill, 1995.

[13] The Network Simulator (NS2), <http://www.isi.edu/nsnam/ns/> 25.08.2010.

 ISBN: 978-1-61208-109-0 

# Equilibrium Strategies for Interference Free Channel Access in Cognitive Radio based WRANs

Swastik Brahma and Mainak Chatterjee
Electrical Engineering and Computer Science
University of Central Florida
Orlando, FL 32816
{sbrahma, mainak}@eecs.ucf.edu

***Abstract*—In this paper, we study how cognitive radio based wireless regional area networks (e.g., IEEE 802.22 networks) can adapt themselves so that they can co-exist with each other. These networks opportunistically accesses and uses under–utilized bands and relinquishes them when the primary user of that band initiates transmission. Such networks, deployed and operated by competing wireless service providers have to self-coexist among themselves by accessing different parts of the available spectrum. Since there is no coordination among the networks in accessing the spectrum bands, they have to adapt such that interference from neighboring networks is minimum. When interfered, a network can adopt either one of two choices– *switch* to a new band hoping to find a non-interfering band, or *stay* with its current band hoping that the interfering network(s) will move away to a new band. Using game theory, we model the spectrum band switching process as an 'infinitely repeating' game where the aim of each network (player) is to minimize its cost of finding a clear channel. We first explore the pure strategy solution space for the game and show that a pure strategy equilibrium, though possible, is infeasible to implement in reality. Thus, we further explore the mixed strategy space and propose a mixed strategy Nash equilibrium among the networks. We analyze the game for the 2-player case, where a network is in interference with only one other network. We also provide hints on how to obtain the equilibrium for the $n$-player game.**

***Keywords*-IEEE 802.22 networks, self-coexistence, game theory, mixed strategy.**

## I. INTRODUCTION

Recent experimental studies that have conclusively shown that licensed radio spectrum is highly under–utilized and that the usage is space and time dependent [17]. In order to take advantage of the spectrum availability due to the analog to digital transition [19], the FCC in the United States has defined provisions to open the sub–900 MHz TV bands for unlicensed services. However, it is mandated for the unlicensed devices to detect and avoid interference with the licensed users in a timely manner [16]. Cognitive radio (CR) based IEEE 802.22 [15] is a wireless regional area network (WRAN), that is targeted to provide a solution to this problem [18]. The aim of IEEE 802.22 is to use spectrum bands dynamically through incumbent sensing and avoidance. For this reason, much of the standard of the IEEE 802.22 is based on cognitive radio. The basic operating principle relies on the cognitive radio being able to sense whether a particular band is being used and, if not, utilize the spectrum without interfering with the transmission of other users (primary incumbents). The elements in these networks (i.e., cognitive radio enabled devices) continuously perform spectrum sensing, dynamically identify unused spectrum, and operate in the spectrum band when it is not used by the primary incumbent radio systems. Upon detecting incumbents, cognitive radio enabled devices are required to switch to another channel or mode. This entails the need of cognitive radio techniques not only to detect the presence/absence of incumbent signals but also to cater to the more important issue of *self-coexistence among the 802.22 networks*. In a typical deployment, multiple 802.22 BSs and CPEs may operate in the vicinity of each other where they compete with each other for grabbing as much spectrum as possible. Different from other IEEE 802 standards where self-coexistence is not a problem, it is so for these networks.

In this paper, we construct an "infinitely repeated" game for the networks (players) for accessing spectrum in an interference free manner where the players always believe that there is some chance the game will continue to the next period. We analyze both pure and mixed strategy solution space for the game and show that a pure strategy equilibria, though possible, is infeasible to implement in reality. We propose a mixed strategy Nash Equilibrium among the players (networks), which does not require negotiation messages to be exchanged between the players.

There are several advantages to taking a game theoretic approach: such model works in a distributed manner where a centralized allocating mechanism is not needed thus making the system scalable. Being rational entities in the game, each of the IEEE 802.22 networks tries to maximize its own payoffs (In our case, minimize the cost of finding a clear spectrum band subject to constraints on resource usage). Furthermore, negotiation messages does not need to be exchanged among the networks. Thus, our solution abides by the assumption that a network does not have any

This research was partially sponsored by ITT Advanced Engineering & Sciences.

 ISBN: 978-1-61208-109-0 

information about the other networks.

The rest of the paper has been organized as follows. In section II we provide a brief discussion of the related works on dynamic spectrum access. Section III formulates the self-coexistence problem as a dynamic channel switching game. The game is analyzed in detail in Section IV, exploring both pure and mixed strategy spaces and their corresponding equilibria. Conclusions are drawn in the last section.

## II. Related work

As far as dynamic spectrum sensing and access are concerned, there is an emerging body of work that deal with different decision making aspects, issues and challenges in cognitive radio network setting. Energy detection have been largely used in [2], [4] to monitor primary spectrum usage activity. Spectral correlation based signal detection for primary spectrum sensing in IEEE 802.22 WRAN systems is presented in [5]. Signature-based spectrum sensing algorithms are presented in [1] to investigate the presence of Advanced Television Systems Committee (ATSC) DTV signals. In a similar effort, sequential pilot sensing of Advanced Television Systems Committee (ATSC) DTV signals is carried out in [9] to sense the primary usage in IEEE 802.22 cognitive radio networks. In [6], a novel frequency sensing method is proposed known as dynamic frequency hopping (DFH). In DFH, neighboring WRAN cells form cooperating communities that coordinate their DFH operations where WRAN data transmission is performed in parallel with spectrum sensing without interruptions. The aim here is to minimize interrupts due to quiet sensing. In [3], a novel metric called Grade-of-Service (GoS) is defined and the trade-off between miss-detection and false alarm is studied for optimizing spectrum sensing performance.

Though most of the above mentioned works focus on primary spectrum usage sensing, however, the issue of self-coexistence among multiple CR networks are not considered. A broad survey on resource allocation in cellular networks using graph coloring mechanisms can be found in [8], [11] and in the references therein. However, most of these works do not consider the dynamic availability of spectrum bands due to the presence of primary users and thus can not be directly applied to IEEE 802.22 network spectrum sharing. In [13], the dynamic channel allocation problem is formulated as graph coloring problem where dynamic channel availability is observed by the secondary users. In [14], spectrum allocation and scheduling problems are studied jointly in cognitive radio wireless networks with the objectives of achieving fair spectrum sharing. However, all channel divisions are treated equally here. In [7], a distributed, real-time spectrum sharing protocol called On-Demand Spectrum Contention (ODSC) is proposed that employs interactive MAC messaging among the coexisting 802.22 cells. However, control signaling is greatly increased through extensive MAC messaging.

## III. Game Formulation

In this section, we formulate the self-coexistence problem as a dynamic channel[1] switching game. We assume that a set of IEEE 802.22 base stations in a region *compete* for one of the $M > 1$ orthogonal spectrum bands not used by primary incumbents. The respective cells for each of the bases stations can be partially or completely overlapped with each other. We refer to such overlapping cells as interfering cells and assume that they can not use the same spectrum band; otherwise QoS of the users of all interfering cells may suffer.

Note that a base station needs to compete only with the neighboring base stations for which there is some overlap[2] for occupying a channel void of interference. If two base stations are more than one hop away from each other, then it is possible for them to occupy the same spectrum band, and thus cannot be said to be competing with each other for occupying a channel [10]. Thus, we consider a base station with its corresponding cell as a rational player and define a set of opponents for each player. The opponent set consists of the the interfering neighbors of the palyer under question. In analyzing the problem, without loss of generality, we can focus our attention on a particular base station with is cell, and analyze the game from its perspective. The same reasoning would apply for other base stations as well.

Let us consider an arbitrary base station and call it player $0$ and let $N$ be the number of its neighbors. Also let $N'$ ($\leq N$) be the number of neighbors with whom player $0$ is interfering (opponents of player $0$), numbered as player $1$, player $2$ through player $N'$. Thus the player set corresponding to the base station under consideration is comprised of players $0, 1, \cdots, N'$.

We investigate the dynamic channel switching game, where the aim of each base station is to capture a spectrum band free of interference from its neighbors, from both pure and mixed strategy perspectives. At the beginning of the game, each base station dynamically chooses one of the $M$ allowable spectrum bands for its operations. If two or more overlapped base stations choose the same spectrum band, then interference will occur and their transmissions will fail. Thus each base station has to pay a price when it experiences interference from its adjoining base stations. Let this cost be $C_I$. Each of these base stations will then have to take decisions regarding whether to stick with the band they have acquired or to switch to a new band in the next stage of the game. When a base station switches to a new channel, it will have to reallocate the new spectrum among its users. This also entails a cost. Let this cost be $C_S$. We assume that $C_S < C_I$, otherwise it does not make sense for

[1]Throughout this paper, we use the words "channel" and "band" interchangeably unless explicitly mentioned otherwise.

[2]Note that, two or more neighboring base stations can operate successfully using different channels.

 ISBN: 978-1-61208-109-0 

a base station to consider switching to a new channel when facing interference.

Next we discuss the two strategies that can be adopted by a base station, viz, the *stay* strategy and the *switch* strategy in their quest to find a clear spectrum band free of interference from its neighbors.

### A. The stage game

If interfered at any stage of the game, player $i$ has the binary strategy set of *switching* to another band (expecting to find a free spectrum band) or *staying* on the current band (assuming the interferers will move away). Using game theoretic notation, the binary strategy set for player $i$ can be represented as:

$$S_i \ = \ \{\text{switch}, \text{stay}\} \tag{1}$$

To generalize, let us assume that a base station (player 0) is interfering with $N'$ of its neighbors (players 1 through $N'$). Let us assume the existence of the strategy set $S_0, S_1 \cdots S'_N$ for the players $0, 1, 2, \cdots, N'$. In this game, at every stage, if player 0 chooses strategy $s_0 \in S_0$, player 1 chooses strategy $s_1 \in S_1$ and so on, then we can describe the set of strategies chosen by all $N' + 1$ players as one ordered $(N'+1)$-tuple, $\mathbf{s} = \{s_0, s_1, \cdots, s_{N'}\}$. This vector of individual strategies is called a strategy profile (or sometimes a strategy combination). For every different combination of individual choices of strategies, we would get a different strategy profile $\mathbf{s}$. The set of all such strategy profiles is called the space of strategy profiles $\mathbf{S}$. It is simply the cartesian product of the vectors $S$ for each base station which can be written as: $\mathbf{S} = S_0 \times S_1 \times \cdots \times S_{N'}$.

Thus we can describe our stage game by the tuple G = $(P, S, C)$, i.e, by a player set $P$, where $P = \{0, 1, \cdots, N'\}$, a space of strategy profiles $S$, where $S = S_0 \times S_1 \times \cdots \times S_{N'}$ and a vector $C$ of von Neumann-Morgenstern utility functions defined over $S$.

### B. The Repeated Game

When a base station is in interference with a subset of its neighbors, it is possible that the base station does not find a clear band in a single play of the stage game $G$ described above. This leads us to the notion of repeated play of the stage game $G$. We assume that the cost incurred by each player from the repeated game is the sum of the cost incurred by the player in each play of the stage game. Since each base station would like to find a clear channel in a minimal amount of time incurring as little cost as possible, our objective is to *minimize* the cost.

Two statements are implicit when we say that in each period we are playing the stage game $G$:

- For each player, the *set of strategies* available to him in any period in the game $G$ is the same regardless of which period it is and regardless of what actions have taken place in the past.
- The payoffs to the players at any stage depend only on the action profile for $G$ which was played in that period, and this stage-game payoff to a player for a given action profile for $G$ is independent of which period it is played.

Before we elaborate on the repeated game strategies, let us first define some notations in the context of the repeated game. We will refer to the strategy of the stage game $G$ which player $i$ executes in period $t$ as $s_i^t$. The strategy profile played in period $t$ is just the $N' + 1$-tuple of each players stage-game strategies:

$$s^t = (s_0^t, s_1^t, \cdots, s_{N'}^t) \tag{2}$$

In order to be able to condition the players' stage-game action choices in later periods upon actions taken earlier by other players we need to define the concept of a history as a description of all the actions taken till the previous period. Formally, the history at time $t$ can be written as:

$$h^t = (s^0, s^1, \cdots, s^{t-1}) \tag{3}$$

In other words, the history at time $t$ specifies which stage-game action profile (i.e., combination of individual stage-game actions) was played in each previous period.

Let player $i$ play the game for T stages. Then we can write players $i$'s strategy for the repeated game as:

$$s_i = (s_i^0, s_i^1, \cdots, s_i^T) \tag{4}$$

We refer to the strategy profile $s$ for the repeated game as the following $N'+1$ tuple profile of players' repeated game strategies:

$$s = (s_0, s_1, \cdots, s_{N'}) \tag{5}$$

*1) Repeated Game Solution Approach:* Whenever we consider a repeated game, we need to define for how long the game will be played. But before we do that we make note of the following two important observations about the game at hand:

1) For a given number of channels $M$, even if a base station (say $n$) finds a clear channel at a stage $t$, it might not correspond to the final assignment for base station $n$. This is because some of its neighbors might still be experiencing interference, because of which they might choose the same channel that base station $n$ is currently residing on at a later stage.
2) The number of available channels $M$ may vary over time since the spectrum usage is time and space variant.

From these two observations, it is clear that a definite ending time (or criteria) for the repeated game can not be inferred. Thus, when a base station (say $n$) faces interference from its neighbors, the decision that base station $n$ takes can not be conditioned by foreseeing future events. It can only be influenced by the past actions taken by its opponents

 ISBN: 978-1-61208-109-0 

(neighboring base stations). Hence, the best that base station $n$ can do, when facing interference, is to play a strategy which is a best response, given base station $n$'s beliefs about the strategies of its opponents, such that its expected cost of finding a clear channel is minimized in the current stage. Since the opponents of base station $n$ also follow this same line of reasoning, the strategies played by the base stations constitute a Nash Equilibrium at every stage of the game. We now consider the following important result:

**Theorem 1:** A sequence of stage-game Nash Equilibrium strategy profiles is also a Nash Equilibrium in the (possibly infinitely) repeated game. More formally, let $\bar{s} = (\bar{s}_0, \bar{s}_1, \cdots, \bar{s}_{N'})$ be a strategy profile for the repeated game. If each of the stage-game strategy profiles $\bar{s}_i$ of $\bar{s}$ is a Nash Equilibrium strategy profile for the stage game, then $\bar{s}$ is also a Nash Equilibrium for the repeated game.

*Proof:* We will prove this by contradiction. Let us assume that $\bar{s}$ is not a Nash Equilibrium for the repeated game. Then for some player $i$ there is an alternative repeated game strategy $\hat{s}_i = (\hat{s}_i^0, \cdots, \hat{s}_i^T)$ which is different, in at least one period, from his part $\bar{s}_i = (\bar{s}_i^0, \cdots, \bar{s}_i^T)$ of $\bar{s}$ such that his payoff in the repeated game is higher, given that everyone else plays their parts of $\bar{s}$. In order that his repeated-game payoff be higher, there must be at least one period $t$ in which his stage-game payoff using $\hat{s}_i^t$ is higher than the stage-game payoff he would get from playing $\bar{s}_i^t$. But if that is true, then $\bar{s}_i^t$ could not have been a part of a Nash-equilibrium strategy profile of the stage game, because some other strategy for player $i$, viz. $\hat{s}_i^t$, would have been better for $i$. This contradicts the hypothesis that every component of $\bar{s}$ is a Nash equilibrium of the stage game. ■

## IV. Game Analysis

With the strategy set and costs defined, the optimization problem is to find a mechanism of switching or staying such that cost incurred can be minimized and an equilibrium can be achieved. We typically assume all the players are rational and pick their strategy keeping only individual cost minimization policy in mind at *every stage of the game*. In this section we analyze a single repetition of the stage game 'G'. We intend to find if there is a set of strategies with the property that no base station can benefit by changing its strategy unilaterally while the other base stations keep their strategies unchanged (Nash equilibrium [12]).

### A. Exploring Pure Strategy Space

We start with the *pure strategy space* played by all the base stations. To simplify investigation of Nash equilibrium with pure strategy space, we consider the game with two players $i$ and $j$ coexisting on one band. The game is represented in strategic form in Table I. Each cell of the table corresponds to a possible combination of the strategies of both players and contains a pair representing the costs of players $i$ and $j$, respectively. Recall that $C_S$ is the cost incurred by a base station when it switches to a new band and $C_I$ in the price paid by a base station experiencing interference in terms of its reduced QoS. Also $C_S < C_I$, as explained before.

| $i \backslash j$ | **Switch** | **Stay** |
|---|---|---|
| **Switch** | $(C_S, C_S)$ | $(\mathbf{C_S}, \mathbf{0})$ |
| **Stay** | $(\mathbf{0}, \mathbf{C_S})$ | $(C_I, C_I)$ |

Table I
PAYOFF MATRIX FOR PLAYERS $i$ AND $j$

As is evident from the table, this game has two pure strategy Nash equilibriums – one corresponding to the strategy profile $(switch, stay)$ with player $i$ choosing to *switch* and player $j$ opting to *stay* and the other corresponding to $(stay, switch)$ with player $i$ choosing to *stay* and player $j$ opting to *switch*. These two cases have been shown in boldface in the table. In both of these cases neither player can reduce his cost by playing a different strategy if the other player plays his part.

However, in reality, neither player can ascertain the strategy to be played by the other player. This inhibits each player from playing a pure strategy. For instance, consider the equilibrium $(switch, stay)$. Player $i$ would *always* choose to *switch* if and only if he knew for sure player $j$ would *always* choose *stay* and vice versa. Similar argument also applies for the other equilibrium, $(stay, switch)$. In practise, all a player can do is to develop a *belief* about the strategy of the other player by forming a probability distribution over the strategies of his opponent based on history of the past stages of the game and act accordingly. This leads us to investigate the mixed strategy solution space of the game.

### B. Exploring Mixed Strategy Space

In the mixed strategy space for the base stations we assign probabilities to each of the strategies in the binary strategy space. We define the mixed strategy space of player $i$ as:

$$S_i^{mixed} = \{(\text{switch} = p_i), (\text{stay} = (1 - p_i))\} \quad (6)$$

where, player $i$ chooses the strategy "switch" with probability $p_i$ and chooses the strategy "stay" with probability $(1 - p_i)$.

Let us first show the 2-player case (two base stations residing on the same band) before we generalize to a $N$-player case.

*1) Mixed Strategy with 2 players:* The 2-player game is represented in strategic form in Table II, with the corresponding mixed strategy probabilities shown.

| $i \backslash j$ | **Switch** $(p_j)$ | **Stay** $(1 - p_j)$ |
|---|---|---|
| **Switch** $(p_i)$ | $(C_S, C_S)$ | $(\mathbf{C_S}, \mathbf{0})$ |
| **Stay** $(1 - p_i)$ | $(\mathbf{0}, \mathbf{C_S})$ | $(C_I, C_I)$ |

Table II
PAYOFF MATRIX FOR PLAYERS $i$ AND $j$

In order to find the mixed strategy equilibria, we need to first find each player's best response correspondence. Player $i$'s best-response correspondence specifies, for each mixed strategy $p_j$ played by player $j$, the set of mixed

 ISBN: 978-1-61208-109-0 

strategies $p_i$ which are best responses for player $i$, i.e, it is a correspondence $p_i^*$ which associates with every $p_j \in [0,1]$ a set $p_i^*(p_j) \subset [0,1]$ such that every element of $p_i^*(p_j)$ is a best response by player $i$ to $j$'s choice $p_j$. The graph of $p_i^*$ is the set of points: $\{(p_i, p_j) : p_j \in [0,1], p_i \in p_i^*(p_j)\}$ Similarly the graph of $p_j^*$ is the set of points: $\{(p_i, p_j) : p_i \in [0,1], p_j \in p_j^*(p_i)\}$.

To find $i$'s best-response correspondence we first compute his expected payoff for the stage game for an arbitrary mixed-strategy profile $(p_i, p_j)$ by weighting each of $i$'s pure-strategy profile payoffs by the probability of that profile's occurrence as determined by the mixed-strategy profile $(p_i, p_j)$ (see Table II):

$$E_i[C] = p_i p_j \frac{M-2}{M-1} C_S + p_i p_j \frac{1}{M-1}(C_S + C_I) + p_i(1-p_j)C_S + (1-p_i)p_j 0 + (1-p_i)(1-p_j)C_I \quad (7)$$

Note that the first two terms in equation (7) both correspond to the strategy profile $(switch, switch)$. This is because when both players $i$ and $j$ chooses to *switch*, there can be two possible associated costs – (i) $i$ and $j$ choose different channels (from $M-1$ possible channels) after switching. The probability of $i$ and $j$ choosing different channels after switching is $(M-2)/(M-1)$ and the cost associated for each player is $C_S$. (ii) $i$ and $j$ choose the same channel (from $M-1$ possible channels) after switching. The probability of $i$ and $j$ choosing the same channel after switching is $1/(M-1)$. The cost associated for each player is $(C_S + C_I)$ since even after switching they still interfere each other.

Player $i$'s minimization problem is:

$$min_{p_i \in [0,1]} E_i[C] \quad (8)$$

Since $p_i$ is $i$'s choice variable, it will be convenient to rewrite equation (7) as an affine function of $p_i$. Simplifying equation (7) we get:

$$E_i[C] = p_i \left[ \frac{M}{M-1} p_j C_I + (C_S - C_I) \right] + [C_I - p_j C_I] = p_i \delta(p_j) + [C_I - p_j C_I] \quad (9)$$

where, $\delta(p_j) = \frac{M}{M-1} p_j C_I + (C_S - C_I)$.

For a given $p_j$, the expected cost function $E_i[C]$ will be minimized with respect to $p_i$ at either:

1) The unit interval's right endpoint (viz. $p_i = 1$) if $\delta(p_j)$ is negative, or,
2) The unit interval's left endpoint (viz. $p_i = 0$) if $\delta(p_j)$ is positive, or,
3) For every $p_i \in [0,1]$ if $\delta(p_j)$ is zero, because $E_i[C]$ is then constant with respect to $p_i$

Let $\delta(p_j)$ be zero at $p_j^*$. To find $p_j^*$, let us equate $\delta(p_j^*)$ to zero: $\frac{M}{M-1} p_j^* C_I + C_S - C_I = 0$. Solving for $p_j^*$ we get:

$$p_j^* = \left(1 - \frac{C_S}{C_I}\right)\left(1 - \frac{1}{M}\right) \quad (10)$$

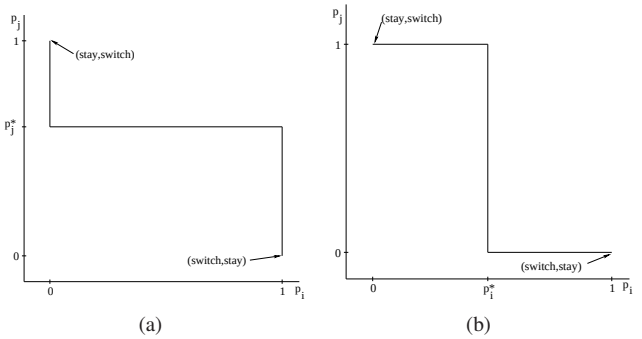



Figure 1. a) Player $i$'s best response for the game; b) Player $j$'s best response for the game.

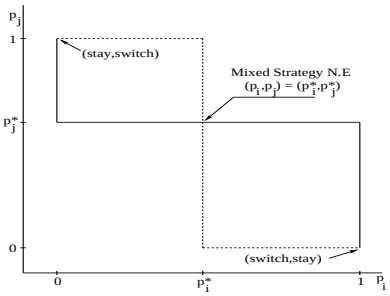



Figure 2. The players best response correspondences and the Nash Equilibrium set.

Now, since we considered that $C_S < C_I$ and $M > 1$, thus $p_j^*$ lies in the interval $(0,1)$. Since $\delta(p_j)$ is an increasing function in $p_j$, player $i$ will choose the pure strategy $p_i = 1$ against $p_j$'s on the interval $[0, p_j^*)$ and the pure strategy $p_i = 0$ against $p_j$'s on the interval $(p_j^*, 1]$. Against $p_j = p_j^*$, player $i$ is free to choose any mixing probability. Player $i$'s best response correspondence is shown in Figure 1(a).

Similarly, player $j$'s expected payoff function is:

$$E_j[C] = p_j \left[ \frac{M}{M-1} p_i C_I + (C_S - C_I) \right] + [C_I - p_i C_I] = p_j \delta(p_i) + [C_I - p_i C_I] \quad (11)$$

Let $\delta(p_i)$ be 0 at $p_i^*$. Then by equating $\delta(p_i^*)$ to 0 we get:

$$p_i^* = \left(1 - \frac{C_S}{C_I}\right)\left(1 - \frac{1}{M}\right) \quad (12)$$

which again lies in the interval $(0,1)$. Since $\delta(p_i)$ is an increasing function in $p_i$, player $j$ will choose the pure strategy $p_j = 1$ against $p_i$'s on the interval $[0, p_i^*)$ and the pure strategy $p_j = 0$ against $p_i$'s on the interval $(p_i^*, 1]$. Against $p_i = p_i^*$, player $j$ is free to choose any mixing probability. Player $j$'s best response correspondence is shown in Figure 1(b).

*The Nash Equilibria are the intersection points in the graphs of player $i$'s and $j$'s best-response correspondences*. This comes directly from the fact that a mixed strategy profile $\{\bar{p}_i, \bar{p}_j\}$ is a Nash Equilibrium if and only if $\bar{p}_i$ is a best response by player $i$ to player $j$'s choice of $\bar{p}_j$ and also $\bar{p}_j$ is a best response by player $j$ to $i$'s choice of $\bar{p}_i$. Figure 2

shows the intersection of the graphs of player $i$'s and $j$'s best response correspondence. We see that the intersection of the graphs of the two best response correspondences contains exactly three points, each corresponding to mixed strategy profile $(p_i, p_j)$: (stay,switch), $(p_i^*, p_j^*)$ and (switch,stay). The first and last of these correspond to the two pure strategy Nash equilibria we identified earlier in Section IV-A. The additional one corresponds to strategy profile $(p_i^*, p_j^*) = ((1 - \frac{C_S}{C_I})(1 - \frac{1}{M}), (1 - \frac{C_S}{C_I})(1 - \frac{1}{M}))$.

Note that, at any stage game, player $i$ does not know what player $j$ will choose $p_j$ to be. Thus what player $i$ does is, based on history, estimates the value of $p_j$. For example, if $i$ observers $j$ switches 7 out of 10 times, then $p_j$ is estimated to be 0.7. Based on this, it takes an appropriate decision depending on where in the interval $((0, p_j^*]$,$p_j^*$ or $(p_j^*, 1])$ the estimated value of $p_j$ lies. Likewise, player $j$ on his part also estimates the value of $p_i$ and takes his decision regarding whether to switch or not. Thus both players $i$ and $j$ play their best response against each other, based on their beliefs of what the other player's strategy is.

*2) Mixed Strategy with $N$ players:* So far, we have shown the equilibrium for the 2-player case. Now, let us provide some insights on finding the equilibrium for the $N$-player game. Let us consider an arbitrary base station and call it player 0. Let player 0 be interfering with $N'$ of its neighboring base stations, numbered 1 through $N'$.

Let $p$ be the 'switching' probability of each one of player 0's $N'$ opponents (i.e., base stations which are interfering with 0). Similar to the arguments for the 2-player case, for player 0 to be indifferent between choosing its two pure strategies, the expected cost when it chooses to 'switch' must be same with that of when it chooses to 'stay'. When player 0 chooses to 'switch', the strategies taken by the opponents of player 0 can result in any $k \in [0, N']$ opponents of player 0 choosing to 'switch'. Let this cost be $E[c_0^{switch}]$. Similarly, when player 0 chooses to 'stay', the strategies taken by the opponents of player 0 can again result in any $k \in [0, N']$ opponents of player 0 choosing to 'switch'. Let this cost be $E[c_0^{stay}]$. As discussed above, $E[c_0^{switch}]$ and $E[c_0^{stay}]$ must be same for player 0 to randomize between his two pure strategies. *Thus, player 0's mixed strategy NE probability of switching will correspond to the roots of the equation obtained by equating $E[c_0^{switch}]$ to $E[c_0^{stay}]$.* This equation can be easily solved by using numerical methods (e.g., bisection method).

## V. Conclusions

In this paper, we use game theory to devise the strategies of cognitive radio based IEEE 802.22 networks such that multiple networks can co-exist even without any coordination among them. A base station's choice of either switching to a new channel or staying with its current channel been modeled as an 'infinitely repeating' game, where each player always believes that there is some chance the game will continue to the next period. The aim of each base station is to minimize its cost of finding a clear channel. Analysis of the game reveals that there exists a pure strategy Nash Equilibria. However, since we cannot assume coordination among the players before a play, implementing such a strategy in reality is infeasible. Thus, we explore the mixed strategy space of the game and propose a solution based on the same. In the mixed strategy space, each player takes an appropriate decision, based on the switching probabilities of its interfering players. The proposed mechanism is distributed in nature without the need for any negotiation messages, thus making the solution scalable.

## References


[1] H.S. Chen, W. Gao, and D.G. Daut, "Signature Based Spectrum Sensing Algorithms for IEEE 802.22 WRAN", IEEE ICC, June 2007, pp. 6487-6492.

[2] G. Ganesan and Y. G. Li, "Cooperative spectrum sensing in cognitive radio networks", in Proc. of IEEE DySPAN, 2005, pp. 137-143.

[3] X. Gelabert, I.F. Akyildiz, O. Sallent, and Ramon Agust "Operating point selection for primary and secondary users in cognitive radio networks" Computer Networks, Volume 53, Issue 8, 11 June 2009, pp. 1158-1170.

[4] A. Ghasemi and E. S. Sousa, "Collaborative spectrum sensing for opportunistic access in fading environments", in Proc. of IEEE DySPAN, November 2005, pp. 131-136.

[5] N. Han, S. Shon, J.H. Chung, and J.M. Kim, "Spectral correlation based signal detection method for spectrum sensing in IEEE 802.22 WRAN systems", ICACT, vol. 3, Feb. 2006, pp. 1-6.

[6] W. Hu, D. Willkomm, M. Abusubaih, J. Gross, G. Vlantis, M. Gerla, and A. Wolisz, " Dynamic Frequency Hopping Communities for Efficient IEEE 802.22 Operation", IEEE Communications Magazine, Volume 45, Issue 5, May 2007, pp. 80-87.

[7] W. Hu, M. Gerla, G.A. Vlantis, and G.J. Pottie, "Efficient, flexible, and scalable inter-network spectrum sharing and communications in cognitive IEEE 802.22 networks", CogART, Feb. 2008, pp. 1-5.

[8] S. Khanna and K. Kumaran, "On wireless spectrum estimation and generalized graph coloring", IEEE INFOCOM, Volume 3, 1998, pp. 1273-1283.

[9] N. Kundargi and A. Tewfik, "Sequential pilot sensing of ATSC signals in IEEE 802.22 cognitive radio networks", IEEE ICASSP, 2008, pp. 2789-2792.

[10] J. Li, C. Blake, D.S.J. De Couto, H. I. Lee, and R. Morris, "Capacity of Ad Hoc wireless networks", ACM Mobicom 2001, pp. 61-69.

[11] L. Narayanan, "Channel assignment and graph multicoloring", Handbook of wireless networks and mobile computing, 2002, pp. 71-94.

[12] J.F. Nash, Equilibrium points in N-person games, Proc. of the National Academy of Sciences, 1950, Vol 36, pp. 48-49.

[13] S. Sengupta, S. Brahma, M. Chatterjee, and S. Shankar, "Enhancements to cognitive radio based IEEE 802.22 air-interface", IEEE ICC 2007, pp. 5155-5160.

[14] J. Tang, S. Misra, and G. Xue, "Joint spectrum allocation and scheduling for fair spectrum sharing in cognitive radio wireless networks" Computer Networks, Volume 52, Issue 11, 8 August 2008, pp. 2148-2158.

[15] IEEE P802.22/D0.1 Draft Standard for Wireless Regional Area Networks Part 22: Cognitive Wireless RAN Medium Access Control (MAC) and Physical Layer (PHY) spec.: Policies and procedures for operation in the TV Bands, 2006.

[16] Federal Communications Commission (FCC), "Notice of Proposed Rule Making", ET Docket no. 04-113, May 2004.

[17] http://www.sharedspectrum.com/measurements/, 2010.

[18] http://www.ieee802.org/22/, 2010.

[19] https://www.dtv2009.gov/, 2010.


 ISBN: 978-1-61208-109-0 

# A Model-Based Approach to the Autonomic Management of Mobile Robot Resources

Adolfo Hernando, Ricardo Sanz
*Autonomous Systems Laboratory*
*Universidad Politécnica de Madrid*
*José Gutiérrez Abascal 2*
*Madrid 28006, Spain*
*Adolfo.Hernando@upm.es, Ricardo.Sanz@upm.es*

Radu Calinescu
*Aston University*
*Aston Triangle*
*Birmingham B4 7ET, UK*
*R.C.Calinescu@aston.ac.uk*

***Abstract*—Mobile robots are increasingly used in applications as diverse as space exploration, rescue missions, automatic floor cleaning in buildings and factories, and mobile surveillance systems. However, one major challenge encountered in the development of applications involving mobile robots is the limited amount of resources that these devices have at their disposal. Effective use of mobile robot resources such as bandwidth is essential for the success of these applications. This paper presents a model-based, utility-driven approach to the autonomic management of mobile robot resources. Starting from a model of the robot resources and of the components that use these resources, the approach builds an autonomic solution capable of optimising a utility function supplied by the robot administrator. This self-optimisation involves adapting the operation mode of the robot components to changes in the amount of resources available and in the environment. The approach is validated through a case study that involves the self-optimising management of bandwidth for a Pioneer 2AT-8 mobile robot.**

***Keywords*-model-based self-X adaptation, mobile robot, utility function, resource management**

## I. INTRODUCTION

As mobile robots are becoming increasingly sophisticated, they acquire the potential to be used in a growing number of applications. For instance, over the past few years our research group at the Universidad Politécnica de Madrid has developed software for the remote control of a Pioneer 2AT-8 robot [1], [2] and its on-board devices. Equipped with this software, the mobile robot offers an increased number of possible configurations. Each of these configurations corresponds to certain on-board devices being active, and is appropriate for different scenarios that the robot may operate in. Furthermore, selecting one or another of these configurations has a different impact on the resource utilisation for the robot as a whole. Some configurations may reduce the time interval that the robot may operate without recharging its batteries, while others may be impossible unless the wireless network used to communicate between the robot and the fixed workstations in the laboratory operate at full capacity.

Taking advantage of these new robot capabilities requires the use of configurations that are dependent on the current state of the robot and its environment, as well as on the objectives of the application that it is involved in. This is a problem widely faced by developers of mobile-robot applications [3], [4].

This paper focuses on the use of autonomic, self-optimising management of robot resources to address the problem described above. An autonomic solution was considered ideal for managing robot resources due to its potential to relieve the human operator from the burden of manually and repeatedly reconfiguring the on-board robot devices. Instead, the operator would be able to specify high-level objectives to be implemented by an *autonomic manager* through monitoring the state of the system, analysing the current state, planning changes required to achieve these objectives and executing these changes. This so-called *monitor-analyse-plan-execute* or MAPE autonomic computing loop is described in detail in [5], [6].

The model-based approach to autonomic resource management introduced in this paper uses the GPAC (General-Purpose Autonomic Computing) platform for the development of autonomic computing applications [7] and defines the objectives for the autonomic management of the robot resources in terms of *utility functions* [8]. The main contributions of the paper include integrating the mobile robot with the GPAC platform, building a model for the self-optimising bandwidth management for a mobile robot, and using it to develop an autonomic resource-management solution based on a set of utility-function autonomic policies. Extensive experiments were carried out to evaluate the effectiveness and overheads of this solution.

The remainder of the paper is organised as follows. Section II describes the mobile robot used in this work, and the GPAC autonomic computing platform. Related work is then presented in Section III, followed by a description of our approach and of the experimental results in Sections IV and V, respectively. Section VI concludes the paper with a brief summary and a discussion of future work.

## II. BACKGROUND

### A. The Pioneer 2AT mobile robot

The work described in this paper was carried out using the Pioneer 2AT-8 mobile robot shown in Figure 1, and

 ISBN: 978-1-61208-109-0 

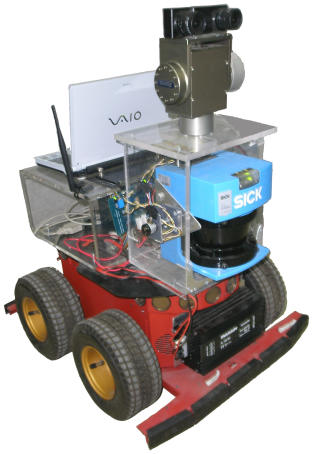



Figure 1. The Pioneer P2AT-8 robot

manufactured by ActivMedia Robotics [9].

The Pioneer 2AT-8 is a skid-steering vehicle with lightweight aluminium body and four pneumatic wheels powered by four reversible DC motors, equipped with high-resolution optical encoders for precise positioning, speed sensing and advanced dead-reckoning. The on-board electronic devices are controlled by a Hitachi H8S microcontroller.

Additionally, the Pioneer 2AT-8 robot has two sonar arrays that provide object detection and range information for collision avoidance. There are two computers on-board the mobile robot, and the experimental setting includes several workstations connected to a local area network and used to control the robot remotely. The on-board computers and the workstations communicate using distributed objects enabled by the Common Object Request Broker Architecture (CORBA) Object-Oriented Middleware, and more specifically the Object Management Group's CORBA 3.1 specification [10].

The control system on the mobile robot enables fine-grained control of the on-board devices through a flexible API implemented by our research group, and which can be accessed via the two computers. This control includes the ability to switch on and off components such as the video camera and the laser SICK[1] or to adjust configuration parameters such as the frames-per-second parameter for the video device. The project described in this paper uses this functionality to adapt the configuration of the robot to changes in the bandwidth of the wireless link between the robot and the workstations used to control it remotely.

The self-adaptation of the robot configuration described above is carried out such as to maximise an analytically defined robot *utility*, subject to not exceeding the available bandwidth. The specification of a utility function for this purpose, and the techniques used to add self-adaptation capabilities to the Pioneer 2AT-8 mobile robot are described in Sections IV and V, respectively.

### B. Autonomic application development with GPAC

GPAC is a service-oriented software platform for the model-driven development of self-* systems [12]. GPAC has been used to develop self-* systems in application domains ranging from autonomic data-centre infrastructure management and dynamic power management to data-storage allocation for military vehicles [13], [14], [15].

The core component of the platform is a reconfigurable autonomic manager capable of augmenting existing IT systems with a MAPE autonomic computing loop. This is achieved using automated code generation techniques to synthesise new software components starting from a model of the IT system that is supplied to the autonomic manager as part of a run-time configuration step [16]. *GPAC system models* describe formally (a) the characteristics of every relevant parameter of the system, including its name, type (i.e., to be `monitored` or `configured` by the MAPE loop) and value domain (e.g., `integer`, `double` or `string`); and (b) the run-time behaviour of the system.

The GPAC autonomic manager supports several types of autonomic computing policies, including action policies, goal policies and utility-function policies [12]. The project described in this paper uses GPAC utility-function policies. To illustrate the specification of this type of autonomic computing policy,[2] and without loss of generality, we will consider policies requiring the optimisation of *multi-objective utility functions* of the form:

$$utility = \sum_{i=1}^{n} w_i \, objective_i, \qquad (1)$$

where the *weights* $w_i \geq 0$, $1 \leq i \leq n$, are used to express the trade-off among the $n > 0$ system objectives. Each objective function $objective_i$, $1 \leq i \leq n$, is an analytical expression of (configurable and monitored-only) system parameters defined in the GPAC system model.

## III. Related work

Utility functions for autonomic systems were originally introduced in [8], and have since been used to build self-managing resource allocation solutions by a number of projects [17], [18]. However, none of the projects that we are aware of has explored the use of utility-based adaptation in managing mobile-robot resources.

[1] A SICK Laser Measurement Sensor (LMS) 200 is an accurate distance measurement sensor that is quickly becoming a staple of the robotics community [11]. The LMS 200 can easily be interfaced through RS-232 or RS-422, providing distance measurements over a 180 degree area up to 80 meters away. The sensor operates by shining a laser off of a rotating mirror. As the mirror spins, the laser scans 180°, effectively creating a fan of laser light.

[2] See [12] for details about the other policy types supported by the GPAC autonomic manager.

 ISBN: 978-1-61208-109-0 

A bandwidth-allocation approach that employs price models to optimise the cost of bandwidth usage is proposed in [19]. This optimization is performed on a network in which there are data packets from different devices that may use different routes. Our problem is different, as it involves a unique route: the wireless network has one access point and a single, multi-device client (the robot). Therefore, our solution optimises the bandwidth usage on this network for the data streams originating from the different, competing devices of the mobile robot.

Other research work addresses the problems of routing in ad-hoc networks (e.g., [20]). These problems are not analyzed in this paper, since we use a managed wireless network, without routing during transmission / reception of wireless data.

## IV. Approach

In our laboratory, the mobile robot was provided with an electronic board that allows to switch on and off the on-board laser range finder, the robotic wrist, the video camera, and the GPS devices programmatically. Note that each of these devices is required only when the robot performs certain tasks, and that they each utilise a specific amount of the bandwidth available between the on-board computers and the fixed control workstations in the laboratory.

### A. Introduction

Our aim was to build a self-managing bandwidth allocation solution enabling the robot to perform its tasks optimally when the available bandwidth is insufficient for the correct operation of all on-board devices. To achieve this, we built a model of the robot that describes the parameters relevant for our application:

- $video \in \{0, 1\}$—a configuration parameter that is 1 if the video camera is switched on, and 0 otherwise;
- $fps \in [0, 200]$—the video frames-per-second rate;
- $width \in \{640, 1280\}$—the video width;
- $rgb \in \{0, 1\}$—the video color space, which is 0 if the color space is Bayer, and 1 if the color space is RGB;
- $laser \in \{0, 1\}$—a configuration parameter that is 1 if the laser SICK is switched on, and 0 otherwise;
- $sonar \in \{0, 1\}$—a configuration parameter that is 1 if the sonar arrays are switched on, and 0 otherwise;
- $brate \in [0, 300]$—a state parameter that specifies the available bandwidth, in Mbits/s;

This model (Figure 2) was used to configure an instance of the GPAC autonomic manager running on one of the workstations in the laboratory. A *manageability adaptor* was implemented using a combination of C# and C++ to allow the autonomic manager to monitor and configure the on-board robot devices.[3]

[3]GPAC manageability adaptors are thin, web-service interfaces that implement a small set of predefined monitoring and configuration operations using the native API of the managed component(s) within a GPAC autonomic system [16].

```
<?xml version="1.0" encoding="UTF-8"?>
<system xmlns="http://www.rcc.com/system"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.rcc.com/system file:/C:
    <name>BandwithControl</name>
    <resource>
        <ID>robot</ID>
        <!-- state parameter section -->
...
        <property>
            <ID>fps</ID>
            <propertyDataType>
                <xs:element name="fps" type="robotfps" />
                <xs:simpleType name="robotfps">
                    <xs:restriction base="xs:double"/>
                </xs:simpleType>
            </propertyDataType>
            <mutability>constant</mutability>
            <modifiability>read-only</modifiability>
            <subscribeability>false</subscribeability>
            <primaryKey>false</primaryKey>
        </property>
        <!-- derived parameter section -->
        <property>
        <!-- configuration parameter section -->
        <property>
            <ID>rgb</ID>
            <propertyDataType>
                <xs:element name="rgb" type="robotrgb" />
                <xs:simpleType name="robotrgb">
                    <xs:restriction base="xs:int"/>
                </xs:simpleType>
            </propertyDataType>
            <mutability>constant</mutability>
            <modifiability>read-write</modifiability>
            <subscribeability>false</subscribeability>
            <primaryKey>false</primaryKey>
        </property>
...
        <!-- operational model section -->
        <property>
    </resource>
</system>
```

Figure 2. GPAC mobile robot model.

In order to use GPAC utility-function policies as described in Section II-B, we also generated an *operational model* describing the relationship between the on-board devices that are switched on and the amount of bandwidth that these devices require. GPAC supports the specification of operational models either as tables whose entries reflect this relationship between parameter values or as analytical Markovian models. This application used the former type of operational model, which was built starting from the equations that relate the bandwidth usage to the configurations relevant for the scenarios in which the mobile robot was envisaged to operate.

### B. Utility functions

Two variants of the utility function in eq. (1) were used. We initially used a simple utility function defined as a linear combination of the video, sonar and laser parameters of the robot, and assuming a fixed value for the 'video frames per second' parameter (i.e., $fps = 3$):

$$utility_1 = (w_1 width + w_2 rgb) video + w_3 sonar + w_4 laser \quad (2)$$

subject to the overall bandwidth usage not exceeding the available bandwidth $brate$:

$$\frac{1024^2}{8} brate \geq video(rgb\ ?\ 3 : 1) width^2 0.75 fps + \\ + 1280\ sonar + 217200\ laser$$

where

$$\begin{aligned} 1280 =& 16 \text{ sonars} * 4 \text{ bytes/sonar} * 20 \text{ Hz} \\ 217200 =& 181 \text{ measures} * 8 \text{ bytes/data} \\ & * 2 \text{ data/measure} * 75 \text{ Hz} \end{aligned} \quad (3)$$

and the notation $x?y : z$ denotes a function that takes value $y$ if its boolean argument $x$ is true and $z$ otherwise.

The weights $w_i$, $1 \leq i \leq 4$, depend on the scenario where the robot is present. This weights are determined by considering the data transmitted over the network link. During daytime, $w_1$ and $w_2$ are positive constants, appropriate for light conditions. In contrast, when the robot operates in a dark environment (e.g., during the night), the weights are set to $w_1 = w_2 = 0$ because using the camera does not contribute to the utility of the robot.

We also experimented with utility functions that take into account the $fps$ frames per second parameter and use exponentials to express the utility of the video camera more accurately:

$$utility_2 = [w_1 A_1(fps) + w_2 A_2(fps) width + \\ + w_3 A_3(fps) rgb] video + w_4 sonar + w_5 laser$$

where

$$A_i(fps) = \left[ \left( 1 - \frac{1}{1 + \exp\left(K_i * (fps - L_i)\right)} \right) - M_i \right]$$

and $K_i, L_i, M_i > 0$ for all $1 \leq i \leq 3$. (4)

This utility functions (Figure 3) is a significant improvement over $utility_1$ because it expresses the fact that very low frames-per-second video has little utility, while the added value of increasing $fps$ grows quickly in the middle of the acceptable range for this parameter. Furthermore, $utility_2$ encodes the fact that increasing $fps$ above a certain value is of limited utility, as the extra amount of video data acquired cannot be processed in real time using the available CPU and memory resources.

Note also that, due to its sigmoid-like shape, $utility_2$ has the additional advantage that it does not suffer from stability problems when the self-optimising system decides the value for the $fps$ parameter. Adding an $w_5 fps$ term to the simpler utility function $utility_1$ was found to suffer severely from such stability problems.

## V. Experimental results

We simulated the network conditions of the robot using real data from the wireless network adapter of a laptop computer similar to the computers on board the mobile robot. Figure 4 depicts three typical experiments using different such data sets for the bandwidth bit-rate and the two utility functions from Section IV-B. This experiments have been chosen to represent, in three small experiments, a broad spectrum of all possible values of the set of available bandwidth values and its variation over time. The first row of diagrams in this figure shows the available bandwidth for the three experiments. The results when $utility_1$ and $utility_2$ were used are illustrated by the diagrams in the second and third row in Figure 4, respectively. Additionally, Tables I and II summarise the results of a larger number of experiments. Note that because the robot utility function can be modified by its administrator at any time, it is not possible to pre-compute the optimal configurations.

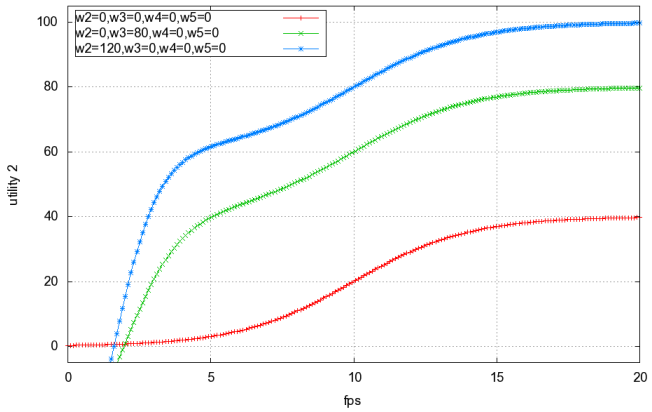



Figure 3. The shape of $utility_2$ as a function of the $fps$ parameter (the weights and the other parameters are fixed: $w_1 = 40$, $K_1 = 0.5$, $L_1 = 10$, $M_1 = 0$ $K_2 = 1.3$, $L_2 = 5$, $M_2 = 0.5$ and $K_3 = 1.05$, $L_3 = 2$, $M_3 = 0.5$.)

### A. Discussion

The results obtained using the two utility-function policies are apparently very similar across the three experiments. When extremely low bandwidth is available (1Mbps), only the sonars are operational, while an increased amount of available bandwidth results in the laser and then the video being switched on. However, there is also a significant difference between the two policies. When $utility_2$ is used, the video is always switched on, and the frames-per-second $fps$ parameter is adjusted to ensure that the mobile robot copes with the available bandwidth. In contrast, recall that $utility_1$ uses a fixed value of $fps = 3$. For low available bandwidth, this $fps$ value is too large, and the video is switched off, bringing the robot into an operating mode in which the range of tasks it can execute is significantly reduced.

 ISBN: 978-1-61208-109-0 

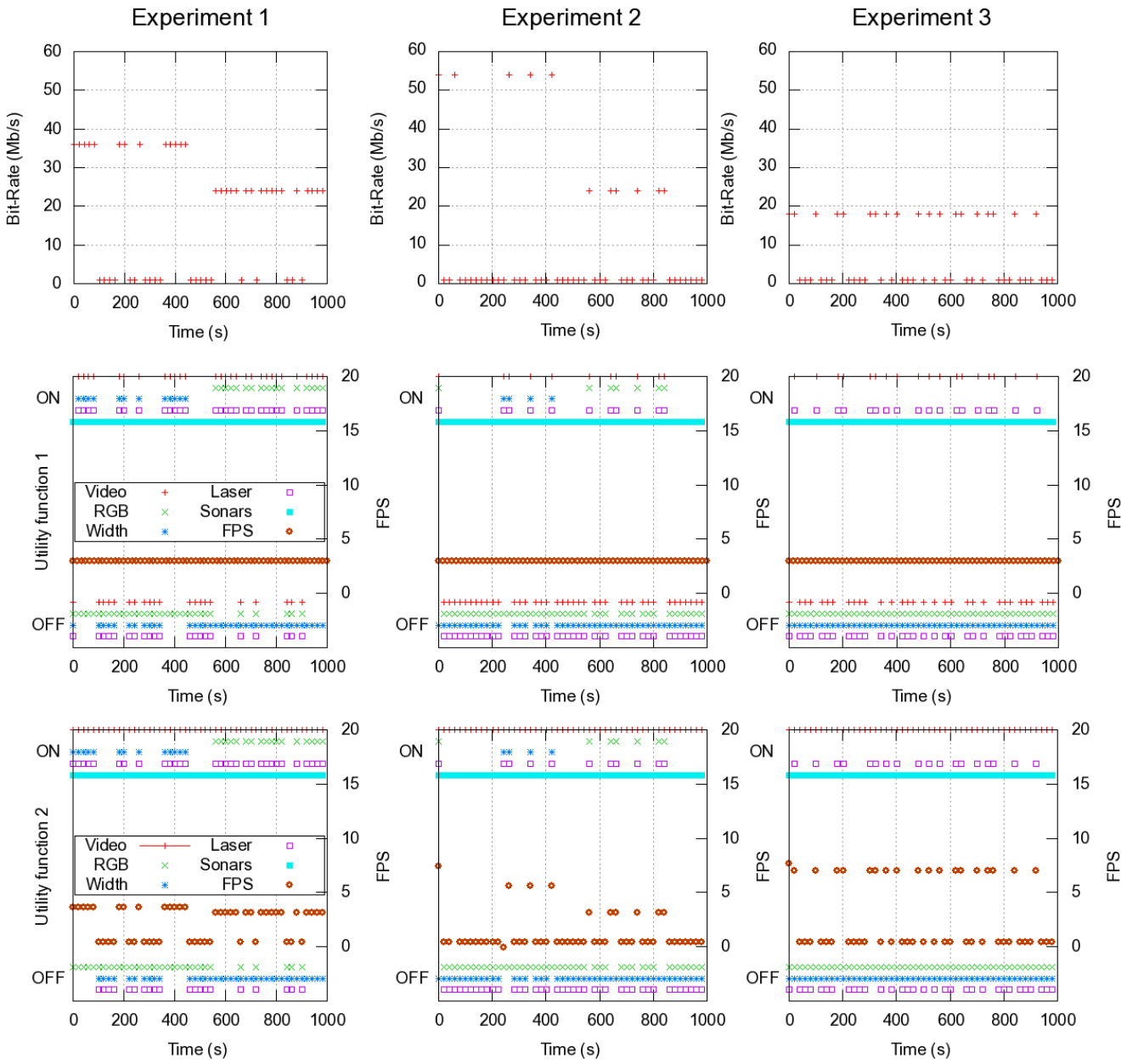



Figure 4. Results of three experiments with two different utility functions—$utility_1$ with $w_1 = 20/1280$, $w_2 = 10$, $w_3 = 20$, $w_4 = 10$; and $utility_2$ with $w_1 = 40$, $w_2 = 120$, $w_3 = 80$, $w_4 = 20$ and $w_5 = 20$. (These weights reflect the expected relevance of the various bandwidth consumers). Width off means $width = 640$, and width on means $width = 1280$.

### B. Overhead

We measured the overhead of the autonomic bandwidth management solution by using the TCP/IP monitoring tool `tcpdump` [21] to keep track of the network traffic between a computer on-board the mobile robot and a remote workstation running the GPAC autonomic manager. Table III shows this overhead both as an absolute figure and as a percentage of the minimum bandwidth available between the mobile robot and the fixed control workstations (i.e., 1 Mbps), when the autonomic MAPE loop is executed once every second. The two rows in the table correspond to two configurations of the wireless connection between the on-board computer and the fixed workstation. The first row presents the overhead when the SOAP_IO_KEEPALIVE flag of the connection was not set. In this case, the underlying gSOAP framework [22] initiated a new connection for each web method call between the autonomic manager and the on-board computer, old connections being closed later.

Table I
RESULTS WITH UTILITY FUNCTION 1

| Bandwidth (Mbps) | Sonars | Laser | Video | RGB | Width | FPS |
|---|---|---|---|---|---|---|
| 1 | on | off | off | – | – | – |
| 2 | on | on | off | – | – | – |
| 6 | on | on | off | – | – | – |
| 9 | on | on | on | off | 640 | 3.00 |
| 12 | on | on | on | off | 640 | 3.00 |
| 18 | on | on | on | off | 640 | 3.00 |
| 24 | on | on | on | on | 640 | 3.00 |
| 36 | on | on | on | off | 1280 | 3.00 |
| 48 | on | on | on | off | 1280 | 3.00 |
| 54 | on | on | on | off | 1280 | 3.00 |

Table II
RESULTS WITH UTILITY FUNCTION 2

| Bandwidth (Mbps) | Sonars | Laser | Video | RGB | Width | FPS |
|---|---|---|---|---|---|---|
| 1 | on | off | on | off | 640 | 0.42 |
| 2 | on | on | on | off | 640 | 0.14 |
| 6 | on | on | on | off | 640 | 1.84 |
| 9 | on | on | on | off | 640 | 3.12 |
| 12 | on | on | on | off | 640 | 4.40 |
| 18 | on | on | on | off | 640 | 6.96 |
| 24 | on | on | on | on | 640 | 3.17 |
| 36 | on | on | on | off | 1280 | 3.66 |
| 48 | on | on | on | off | 1280 | 4.94 |
| 54 | on | on | on | off | 1280 | 5.58 |

Note that this behaviour was noticed thanks to the use of `tcpdump`, so in a second suite of experiments we set the SOAP_IO_KEEPALIVE flag to ensure that the connection between the autonomic manager was reused across multiple web method calls. While this improvement yielded an almost 20% reduction in the bandwidth overhead, the overhead when the default configuration was used (i.e., 3.9% overhead at the lowest possible bandwidth) was already acceptable. Note that for increased accuracy, this overhead can be taken into account in eq. (3).

Table III
AUTONOMIC MANAGER OVERHEAD

| Case | Send bytes/s | Receive bytes/s | Total bytes/s | Overhead (at 1Mbps) |
|---|---|---|---|---|
| No Keepalive | 2847 | 2267 | 5114 | 3.9% |
| Keepalive | 2313 | 1809 | 4122 | 3.14% |

## VI. CONCLUSION AND FUTURE WORK

We integrated a mobile robot and a simulation of this robot with the GPAC autonomic computing framework. This enabled us to build a MAPE autonomic computing loop for the self-optimising management of the bandwidth between the on-board robot devices and the fixed workstations used to control this robot. Experiments were carried out to evaluate the effectiveness and the overhead of the solution.

There are several lines of work we would like to pursue in the future. We are planning to extend the use of the approach described in the paper to other types of robot resources (e.g., power usage). Additionally, we would like to improve the execution of the MAPE autonomic computing loop so that it is executed (and adds to the communication between the on-board robot laptops and the control workstations) only when a "significant" change occurs in the relevant robot parameters.

## REFERENCES

[1] R. Sanz, A. Hernando, C. Martínez, and I. López, "Wrapping a Mobile robot with RT-CORBA," in *VIII IFAC Symposium on Robotic Control (SYROCO'06)*, September 2006.

[2] A. Hernando, "Versión RT-CORBA del servidor Pioneer 2AT-8," Master's thesis, ETSII, Universidad Politécnica de Madrid, October 2005.

[3] Y. Meng, "A Dynamic Self-Reconfigurable Mobile Robot Navigation System," in *Proc. of the 2005 IEEE/ASME Intl. Conf. on Advanced Intelligent Mechatronics*, July 2005.

[4] B. MacDonald, B. Hsieh, and I. Warren, "Design for Dynamic Reconfiguration for Robot Software," in *2nd Intl. Conf. on Autonomous Robots and Agents*, Palmerston North, New Zealand, December 2004.

[5] "An architectural blueprint for autonomic computing," IBM, Tech. Rep., 2003.

[6] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, no. 3, pp. 1–28, 2008.

[7] R. Calinescu, "Model-driven autonomic architecture," in *ICAC '07: Proc. of the Fourth Intl. Conf. on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2007, p. 9.

[8] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das., "Utility Functions in Autonomic Systems," in *First Intl. Conf. on Autonomic Computing*. IEEE Computer Society, 2004, pp. 70–77.

[9] *Pioneer 3 & Pioneer 2 H8-Series Operations Manual, version 3*, ActivMedia Robotics, August 2003.

[10] *Common Object Request Broker Architecture (CORBA/IIOP)*, 3rd ed., Object Management Group (OMG), January 2008.

[11] K. Sevcik, "Interfacing with the Sick LMS-200," web page. [Online]. Available: http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html

[12] R. Calinescu, "General-purpose autonomic computing," in *Autonomic Computing and Networking*, M. K. Denko, L. T. Yang, and Y. Zhang, Eds. Springer, 2009, pp. 3–30.

[13] E. Werkman, B. van Schoonhoven, M. de Jonge, and E. Matthijssen, "Development of autonomic management solutions for the military application domain," in *5th IEEE Intl. Conf. on Engineering of Complex Computer Systems*. IEEE Computer Society, 2010, pp. 14–20.

 ISBN: 978-1-61208-109-0 

[14] R. Calinescu and M. Kwiatkowska, "Using quantitative analysis to implement autonomic IT systems," in *Proc. of the 31st Intl. Conf. on Software Engineering (ICSE'09)*, 2009, pp. 100–110.

[15] ——, "CADS*: Computer-aided development of Self-* systems," in *Fundamental Approaches to Software Engineering (FASE 2009)*, ser. Lecture Notes in Computer Science, M. Chechik and M. Wirsing, Eds., vol. 5503. Springer, March 2009, pp. 421–424.

[16] R. Calinescu, "Reconfigurable service-oriented architecture for autonomic computing," *Intl. Journal On Advances in Intelligent Systems*, vol. 2, no. 1, pp. 38–57, June 2009.

[17] G. Tesauro *et al.*, "A multi-agent systems approach to autonomic computing," in *AAMAS '04: Proc. of the Third Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 464–471.

[18] J. O. Kephart and R. Das, "Achieving self-management via utility functions," *IEEE Internet Computing*, vol. 11, pp. 40–48, 2007.

[19] Y. Qiu and P. Marbach, "Bandwidth allocation in ad hoc networks: A price-based approach," in *IEEE INFOCOM*, 2003, pp. 797–807.

[20] I. N. Kassabalidis *et al.*, "Intelligent routing and bandwidth allocation in wireless networks," in *Proc. NASA Earth Science Technology Conf.*, College Park, MD, 2001.

[21] Lawrence Berkeley National Laboratory (LBNL), "tcpdump," web page. [Online]. Available: http://www.tcpdump.org/

[22] R. A. van Engelen, "gSOAP toolkit," web page. [Online]. Available: http://gsoap2.sourceforge.net/

 ISBN: 978-1-61208-109-0 

# An Overall Process for Self-Adaptive Pervasive Systems

Antonio Bucchiarone*, Annapaola Marconi*, Marco Pistore*,
Stefan Föll†, Klaus Herrmann†, Christian Hiesinger†, Srdjan Marinovic‡
*FBK-IRST, Trento, Italy
{bucchiarone,marconi,pistore}@fbk.eu
†Universität Stuttgart, Germany
{stefan.foell,klaus.herrmann,christian.hiesinger}@ipvs.uni-stuttgart.de
‡Imperial College, London, UK
srdjan@imperial.ac.uk

***Abstract*—Self-adaptive pervasive systems often implement adaptation in a centralised manner, where one component holds all the necessary knowledge to identify when and how the system needs to adapt. In self-adaptive pervasive systems, composed of autonomous components with different authorities (such as security, distribution, etc.), this approach cannot be implemented as composing a centralised knowledge is not feasible and it also obstructs the system's ability to dynamically change its components. A simple alternative would be to allow each component to adapt independently but this can quickly give rise to conflicts, race conditions and oscillations between multiple independent adaptations. To avoid these problems, we propose to coordinate individual adaptations so that each component's adaptation goals are satisfied. Each component proposes an adaptation which is reviewed by other components who may propose their own adaptations that they may need to do. This continues until a complete adaptation plan is agreed upon. In cases where certain individual adaptations conflict with some components' goals, components are instructed to seek alternative proposals. The *Adaptation Manager* component is in charge of the negotiation process and it also has the authority to resolve certain conflicts between adaptations. Our approach is evaluated in the context of pervasive workflow systems where the failure probability and execution times are assessed.**

***Keywords*-Pervasive Systems; Adaptation; Workflows**

## I. INTRODUCTION

Self-adaptive pervasive systems operate in environements characterised by a high degree of dynamic behaviour due to frequent changes in the environment and frequent changes of the environment in which they operate [1]. As a running use-case study of adaptive pervasive systems, this paper will adopt the system described in the ALLOW project [2], that focuses on modelling and building human-centric pervasive systems. The project is also heavily investigating the applications of workflows as a new programming paradigm for these systems. More precisely, adaptable pervasive systems (APSs) are modeled using Adaptable Pervasive Flows (APFs) [3], [4], [5] and these flows represent an extension of traditional workflow concepts [6] which make them more suited for adaptation and execution in pervasive execution environments.

Principal application adaptations are executed by replacing or rearranging tasks in a flow such that the flow's goals can still be fulfilled despite the fact that various failures and problems have occurred. Beyond that, a number of other adaptation aspects need to be controlled while a flow is running such as: the adaptation of user interfaces, security adaptation, and flow distribution adaptation. The adaptation strategies for these different aspects of the flow execution are stored and executed by the components in control of these aspects, such: security manager, distribution manager and so forth. Having each component adapt its own concerns independently of the rest of the system, makes the adaptation much more robust since the adaptation knowledge and strategies are decentralised. This in turn makes the components loosely-coupled and the system more robust in face of component failures. However, this also introduces a problem of coordinating these different independent adaptations to avoid conflicts, oscillations and race conditions among their adaptation actions and goals.

The main contribution of this paper is to propose a novel approach for coordinating a set of autonomous components that can adapt different execution aspects of a common pervasive system. Our solution allows the components to retain their autonomy and authority over the execution aspects that they are overseeing, while undesirable effects such as conflicts, oscillations and race conditions are avoided. We have implemented this approach for the pervasive flow systems and we use this implementation to validate our ideas. We refer to this approach as the Overall Adaptation Process (OAP). The OAP approach is based on two main principles: 1) every aspect publishes the changes it is going to make to the system, 2) other aspects say whether they agree with the changes and whether they themselves need to make further changes to accommodate the original proposed ones. In order to coordinate this negotiation loop between different components we introduce the *overall adaptation manager* whose internal logic guides the negotiation to an overall agreement.

The rest of the paper is structured as follows: Section II analyses the requirements that the proposed adaptation

 ISBN: 978-1-61208-109-0 

approach, specified in Section III, ought to meet. Section IV discusses testing and validating the proposed approach, while the last two explore the relevant related works and conclude the paper.

## II. Problem Analysis

The nature of pervasive systems defines extensive requirements for the design of self-adaptive systems. First, pervasive systems are becoming increasingly complex and are required to manage a large set of different aspects of the system. Each of these heterogeneous aspects considers a bounded part of the overall system and is associated with a dedicated goal for execution. This goal is defined by means of *acceptance criteria*. Second, adaptation is considered as a key principle to deal with the dynamics found in pervasive environments, which results from unforeseen changes in the system that may violate each aspect's acceptance criteria. Therefore, adaptations are necessary to bring the system back into a state that preserves the individual criteria of each aspect. Due to the above characteristics, the overall complexity of the system is only manageable by avoiding a single, centralized point of control. The reduction of complexity can be achieved through a loosely-coupled system design, where the aspect-specific adaptation strategies are independently encapsulated inside each aspect. Thus, aspects can be flexibly composed to suit the characteristics of different environments, i.e., some aspects may not be present in each setting.

However, this model of independently acting aspects also elevates certain issues. Although each aspect is only concerned with a part of the system, harmful side-effects resulting from adaptation may occur. In particular, the adaptation of one aspect may easily push the system into a state that triggers adaptations in other aspects. This causes the following problems:

**Conflicts:** A conflict arises if two or more aspects react to a relevant change by issuing contradictory adaptations concurrently. Contradictory adaptations are adaptations that cannot be applied in combination since this would violate the acceptance criteria of the system.

**Oscillations:** An oscillation can occur if conflicting adaptations are issued in sequence: If one aspect $as_1$ receives an event that violates its internal acceptance criteria it issues an adaptation which in turn violates the acceptance criteria of one or more other aspects. These aspects react by issuing adaptations to correct this violation, which in turn violates the criteria of $as_2$. This can lead to a long series of mutual adaptations without ever reaching a system configuration that satisfies all aspects.

**Race conditions:** Race conditions occur if two or more aspects apply adaptations and the final outcome depends on the order in which they complete and apply their individual adaptations. In particular, if we have aspects $as_1$ and $as_2$ issuing adaptations, the order $as_1, as_2$ may lead to a system configuration that satisfied all acceptance criteria while the order $as_2, as_1$ may result in a conflict or oscillation.

In order to avoid these problems, a coordinated way of interaction among the aspects is required. The goal is to restrict the freedom of each individual aspect in order to avoid the negative effects detailed above while still preserving as much of the autonomy as possible. The principal question that we are tackling in this paper is therefore: *How can loosely coupled autonomous adaptation aspects be coordinated while their autonomy is preserved*?

## III. Cross-Aspect Adaptation

The novel approach for cross-aspect adaptation proposed in this paper is based on the insight that aspects need to coordinate at an early stage of adaptation before applying changes to the system. For this reason we advocate the introduction of an adaptation process split into phases of *adaptation negotiation* and *adaptation enactment*. Adaptation negotiation denotes the process of aspect interaction with the goal of finding an acceptable adaptation that all aspects agree on. Adaptation enactment refers to the physical adaptation of the system, based on the agreement achieved by the negotiation process before. Both phases are usually strongly integrated into one indivisible process in traditional systems. However, the decoupling of these constituent parts of our adaptation process enables adaptation aspects to reason over the effects of possible adaptation and influence other aspects in decision making before the actual change of the system.

In the next subsections, we describe our solution that facilitates coordinated adaptation among the aspects in the system. For this purpose, we first introduce in Section III-A the adaptation framework, which presents the necessary elements and their relationships for building self-adaptive systems according to our approach. Subsequently, we use these elements to define our overall adaptation process in Section III-B.

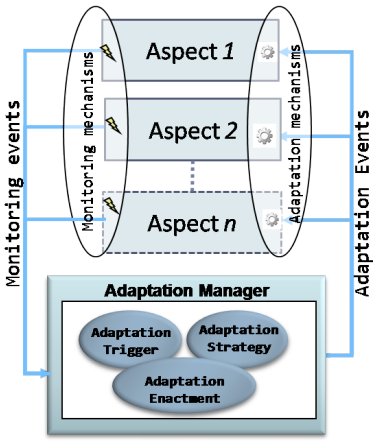



Figure 1. Overview of Cross-Aspect Adaptation Framework

### A. Adaptation Framework

The adaptation framework serves as a blueprint for the design of composed adaptive systems. It defines a unified view on the elements of a self-adaptive pervasive system as illustrated in Figure 1. The elements are *Adaptation Aspects* representing the self-contained entities executing adaptations in the system and the *Adaptation Manager* that represents the control instance of a coordinated adaptation process.

An adaptation aspect covers a well-defined part of the pervasive system subject to dynamic adaptation. The *model* of an adaptation aspect $as_i$ is expressed as a set of public variables $V(as_i)$ modelling the system properties relevant for adaptation. An assignment of the variables in $V(as_i)$ is called a *configuration* of the aspect and denoted as $v_{as,i}$.

Each aspect represents an autonomous part of the system, which monitors changes of the system relevant to its acceptance criteria. For this purpose, each aspect provides a set of *monitoring mechanisms* to detect events that provide information on the execution environment. These events are used to inform the Adaptation Manager about aspect-specific problems that demand an adaptation. Moreover, each adaptation aspect has a set of *adaptation mechanisms* that can be used to enact adaptations physically in the environment. The concrete adaptation mechanisms are aspect-specific and depend on the functionality provided by each individual aspect. While aspects change the system according to their adaptation needs, they are not required to know or understand the direct effects of their own adaptations on other aspects. This preserves their fundamental level of autonomy as aspects just have to implement their internal adaptation logic and know their own acceptance criteria.

Aspects are able to communicate with each other based on a common negotiation language. The negotiation language of aspects $AS_{1,n} = \{as_1, as_2, .., as_n\}$ is based on all variables $V(AS_{1,n}) = V(as_1) \cup V(as_2) ... \cup V(as_n)$ shared among them. The variables are used to communicate the changes an aspect is going to make for adaptation. An assignment of all variables is formally referred to as $v_{AS_{1,n}}$ and indicates a desired *system configuration*, which may result from a series of adaptations of a set of adaptation aspect. The change of an aspect's configuration has consequences on all aspects that are influenced or depend on its shared properties. Thus, the coordination of several aspects is based on the exchange of system configurations, allowing all aspects to inspect and influence the future adaptation of the system. As the exchanged system configurations describe hypothetical system states under negotiation, they can be seen as *proposals* of an overall adaptation that involves more than one aspect for adaptation.

In addition to the adaptation aspects, our framework also introduces the *Adaptation Manager* (AM) component whose purpose is to coordinate multiple independent adaptations. This introduction is motivated by the fact that a well-structured way of interaction among the aspects is required, which needs to be controlled by a mediator. The Adaptation Manager reacts to monitoring events arriving from the adaptation aspects. Based on the incoming events, the AM infers an adaptation trigger $a_t$ that characterizes the cause of adaptation. The trigger may be the result of a single monitoring event, as well as be discovered from the correlation of multiple events coming from different adaptation aspects. The AM acts upon a set of adaptation strategies, each of which defines a coordination process for specific adaptation triggers. Formally, an adaptation strategy $a_s$ is a tuple $(a_t, a_p, a_e, p)$ where $a_t$ denotes the adaptation trigger the strategy is devised for, $a_p$ is the adaptation plan, $a_e$ describes the adaptation enactment, and $p$ is the priority associated to the strategy. The adaptation plan $a_p$ associated with an adaptation strategy $a_s$ defines the procedure of how to negotiate among the aspects in order to react appropriately on an adaptation trigger. For this purpose, the adaptation strategy reflects the dependencies among the adaptation aspects. The dependencies of aspects define the order, in which aspects can make their proposals of desired adaptations. For example, if the adaptation manager selects a strategy with the associated plan $\langle as_i, as_j, as_k \rangle$, it means that the aspects are contacted in the order $as_i, as_j, as_k$. Similarly, the adaptation enactment $a_e$ is a list of aspects and captures the order in which different aspects should be invoked to enforce the adaptation. This allows complex adaptations, where not only the agreement for finding an adaptation, but also the implementation of adaptations requires a coordinated approach.

### B. Overall Adaptation Process

The process of overall adaptation defines an interaction protocol in which the adaptation aspects iteratively take turns in a well-defined order and propose adaptations. In this way, adaptations are not occurring concurrently but they are serialized. Each aspect thus does not base its adaptation on the actual current system configuration but on the hypothetical system configuration that would exist if the previously proposed adaptations had been applied. This process is coordinated by the Adaptation Manager and consists of four individual phases, which are described in detail subsequently:

**Initialization** *(Phase 0)*: During the system execution, the set of adaptation aspects publishes monitoring events. The AM, using a correlation function $C$, infers an adaptation trigger $a_t$ to launch the overall adaptation process. In order to react appropriately on $a_t$, the $AM$ selects all adaptation strategies $\{\langle a_t, a_p, a_e, p \rangle\}$ which are devised for this trigger. Among this set, the strategy with the highest priority $p$ is chosen for execution. Each aspect listed in the adaptation plan $a_p$ will participate in the overall adaptation process. Let these aspects be denoted as $AS_{1,n} = \{as_1, as_2, ..., as_n\}$ in

 ISBN: 978-1-61208-109-0 

the following. The initial configuration of the system before negotiation is thus given by the values of all variables in $V(AS_{1,n})$ found in the running system at the time a trigger is raised. The goal of the OAP is then to negotiate a future configuration of the system, so that the individual requirements of each aspect can be met.

**Negotiation phase** *(Phase 1)*: In the negotiation phase a consensus protocol is executed that involves interactions of the AM with the adaptation aspects. The AM controls the interactions according to the adaptation strategy and collects the adaptation proposals from the aspects. An adaptation proposals represents a hypothetical aspect configuration that would be the result of an adaptation action. It is important to note that the aspect configuration is merely virtual and describes the changes of a desired adaptation in an understandable way for the remaining aspects. This creates the opportunity for aspects to inspect the objectives of other aspect before their physical enactment. For this purpose, each aspect $as_i$ provides a uniform interface $propose_{as_i}$ that can be invoked by the AM during the negotiation phase:

$$propose_{as_i} : (v_{AS_{1,n}}, a_t) \mapsto v'_{AS_{1,n}}$$

Each aspect participates by suggesting an adaptation, which in turn provides the input for aspects consulted subsequently. The proposal is made for a specific adaptation trigger and based on the proposals made by other aspects which came before. This process ensures to exchange adaptation proposals in a way, which takes the dependencies among the aspects into account. This iterative procedure results in an evolution of the future system configuration from the first version, which is the initial proposal of first aspect, to the final one, where each aspect has stated its proposal. This procedure can be seen as a concatenation of the proposal functions: $v'_{AS_{1,n}} = propose_{as_n}(...propose_{as_2}(propose_{as_1}(v_{AS_{1,n}}, a_t), a_t), a_t)$. The proposal phase is completed after the last aspect $as_n$ from the set of participating aspects has contributed its proposal. The result is a new hypothetical system configurations, that needs to be validated for enforcement. As the AM forwards a proposals from one aspect to another, the aspect have not to be aware of each other directly.

**Decision phase** *(Phase 2)*: After each aspect has taken its turn, an agreement needs to be reached whether one coherent overall adaptation (result of all the aspect-specific adaptations) was found. For making this decision, each aspect needs to rate the outcome of the proposal phase with regards to its individual acceptance criteria. As the rating of an aspect may be influenced by proposals made *before* and *after* an aspect's own turn, the decision making is reached in a separate phase that allows all aspects to see the final negotiated configuration $v'_{AS_{1,n}}$. For rating, each aspect may choose from a set of aspect-specific *verdicts* $r_{as_i} = \{r_1, .., r_n\}$ that are also known to the AM. An aspect can communicate a quality measure via a verdict which allows the AM to infer the satisfaction of an aspect with the adaptation proposal. The rating is calculated using the function $rate_i$ of each aspect that can be formalized as:

$$rate_i : v'_{AS_{1,n}} \mapsto r \in r_{as_i}$$

so that the overall rating $r_{overall} = (r_1(v'_{AS_{1,n}}), ..., r_n(v'_{AS_{1,n}}))$ can be expressed as the tuple of all the aspects' verdicts. The AM inspects the overall rating and determines whether one coherent overall adaptation was found. For this purpose, the AM holds decision rules that determine if the proposal can be accepted:

$$decide : r_{overall} \mapsto \{accepted, rejected\}$$

Thus, the AM represents the final authority for enforcing adaptation decision. If the overall rating is not sufficient for enforcing an adaptation, the AM will start another negotiation phase as described before such that the solutions proposed in the previous round are avoided by the aspects who proposed the first initial proposal. In each loop, all aspects are required to *relax their adaptation criteria*. Such a relaxation is realized by applying a less restrictive threshold to some quality criteria an aspect uses internally to judge whether a given system configuration is sufficient. By means of criteria relaxation, the set of possible system configurations and thus the probability of finding valid overall adaptations is increased. The termination of the negotiation loop is ensured by a defined strategy associated with the adaptation plan, e.g., restricting the number of negotiating cycles.

**Phase 3 – Enactment phase** *(Phase 3)*: If a valid adaptation is found in phase 2, the hypothetical configuration on which the aspects have agreed needs to be physically implemented in the system. This again motivates a coordinated approach among the aspects, since an aspect may already require the configuration of another aspect to be physically existent in order to implement its target configuration. Therefore, the aspects again take turns in a well-defined order. The order is defined by the enactment $a_e$ as part of the selected adaptation strategy $a_s$. Thereupon, each aspect implements its part of the overall system configuration. The required actions to enforce an adaptation for an aspect $as_i$ are directly derived from the target configuration $v_{AS_{1,n}}$ of the agreed overall system configuration.

## IV. EVALUATION

To evaluate the feasibility of our proposed overall adaptation approach, we applied our proposal in the context of the EU FP7 ALLOW project [4]. This project focuses on human-centric pervasive applications. Applications are

 ISBN: 978-1-61208-109-0 

specified as adaptive pervasive workflows. To execute a workflow successfully, a workflow must adhere to a set of functional and non-functional constraints. In the ALLOW project, three different adaptation aspects are considered: 1) the *Flow Security Aspect* which ensures that the execution of a workflow is according to specified security policies. 2) the *Flow Structure Aspect* which may modify the structure of a workflow in order to meet the specified goal of the workflow. 3) the *Flow Distribution Aspect* which ensures that the execution of a workflow complies to specified performance criteria.

Without coordination adaptation the problems mentioned in Section II may occur. Assume, for example, the *Flow Security Aspect* replaces untrusted services with trusted ones. The new set of services does not suite the performance criteria of the *Flow Distribution Aspect*. Thus, the *Flow Distribution Aspect* replaces the trusted services with untrusted ones that have a better performance which would in turn trigger the *Flow Security Aspect* again. Consequently, the system would oscillate.

To measure the effectiveness of our approach, we compared a system with adaptation manager controlling the overall adaptation process to a system without adaptation manager. We define a system without adaptation manager as a system without any coordination between the individual aspects. We conducted two sets of experiments. In the first set, we assumed a workflow engine aborts the execution of a workflow if a task blocks the execution because it cannot be executed without violating the acceptance criteria of the individual aspects. In this set, we measured the number of completely executed workflows.

In the second set, we assumed a workflow engine waits until a task can be executed without violating the acceptance criteria of the individual aspects. In this set, we measured the average time required to completely execute workflows.

Our evaluation is based on simulation. In each simulation run, we generated a workflow and executed the workflow in each system under identical conditions. This means, that the environment a workflow is executed in generates the same events at the same point in time. In the simulation, we measure time in simulation cycles.

The environment influencing the execution of our workflows consists of a network (simulating the electronic world) and a context system (simulating the real world). Both, network and context system may generate events at arbitrary points in time during a simulation run. These events may result in an adaptation of the currently executed workflow. Network events give information about changes in the availability or the trust of services provided by the network. Events of the context system indicate that the goal of a workflow is endangered due to real-world events.

This information is used by our aspects as follows. The *Flow Security Aspect* ensures that all services which are bound to the tasks of a workflow are according to the specified security policies. If this is not the case, the *Flow Security Aspect* replaces untrusted services with trusted ones if possible. The *Flow Distribution Aspect* binds the fastest available service in order to improve the execution time of a workflow. Whenever an event endangers the goal of a workflows, the *Flow Structure Aspect* tries to replace some tasks of the workflow with a predefined set of tasks. Hereby, the *Flow Structure Aspect* can chose from several predefined sets.

The probability for changes in the network or for context events is defined by a failure probability. In each simulation cycle, each service changes its state according to this failure probability. Similarly, context events are generated with this probability in each simulation cycle.

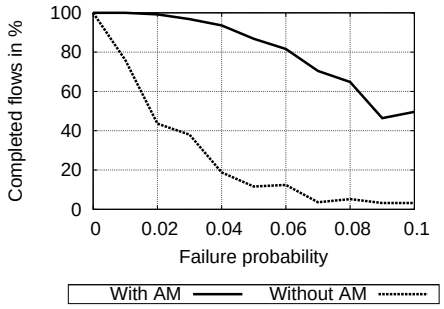



Figure 2. Engine aborting in case of a blocking task

Figure 2 shows the evaluation results for the first set of experiments. It can be seen that in the system with adaptation manager more workflows are executed completely than in the system without adaptation manager. This is due to fact that the system with adaptation manager is able to resolve conflicts and oscillations by means of its negotiation process. However, if the failure probability increases, the probability rises that no solution exists in the whole solution space. Thus, the percentage of successful executions decreases with an increase in the failure probability.

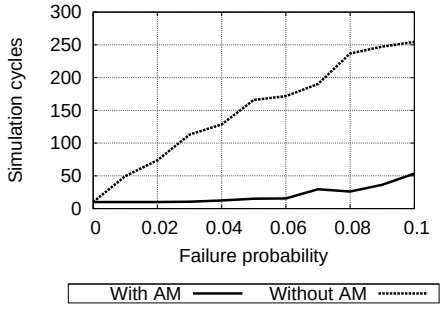



Figure 3. Engine waiting for tasks to complete

In Figure 3, the results for the second set of experiments are depicted. It can be seen that the number or simulation

 ISBN: 978-1-61208-109-0 

cycles is much higher in the system without adaptation manager. This is because without coordination the aspects are not able to derive compromises and, thus, have to wait for a configuration that by chance fits all acceptance criteria. Again, as the failure probability increases, it may happen that no solution for a problem can be found. In this case, also the system with adaptation manager has to wait and, thus, the average time required to execute a flow increases.

## V. Related Work

The motivations for, and a notion of coordinated adaptation has been investigated by Cheng et al. [7]. They have argued that independent adaptations must not introduce unwanted effects through which the system can be put into an unsafe state. These unwanted behaviours can be oscillations, races, deadlocks and so forth [8]. They propose that components should coordinate their evaluation metrics to reach a consensus, but they do not have a strategy to deal with situations when such consensus cannot be reached.

CARISMA system [9] explores bidding on a proposed set of adaptation policies as a way to adapt the whole system. When a set of proposals is collected every agent places a sealed bid for each proposal. The proposal with the highest sum of bids, wins. The agents implement a utility function that decides how much they are willing to bid on a particular proposal. The main difference with our work is that CARISMA has no further rounds, where agents would be able to revise their utility functions. This is a crucial requirement for our system since the workflow adaptation component can revise its adaptation plan based on a set of available and trusted services.

In [10] each application consists of different functional levels where each level has its own evaluation metrics. The model employs an independent component that monitors the performance of individual layers. If certain constraints are not satisfied, this monitor invokes different adaptation mechanisms in affected layers. The main difference with this approach is that the presented overall adaptation manager runs a negotiation loop.

## VI. Conclusion

Self-adaptive pervasive systems that are composed of cooperating and autonomous components, typically employ each component to control and monitor a specific application execution aspect. Each of these components usually is capable of self-adapting a particular part of the underlying pervasive system or application in order to meet its own goals.

To avoid problems like conflicts, race conditions and oscillations, this paper proposes a solution based on a negotiation loop that attempts to find one adaptation that satisfies all components' individual constraints and needs. The negotiation loop provides a coordinated way for individual components to exchange adaptation proposals and infer whether such proposals break their individual requirements. This process is coordinated by a special component, the *Adaptation Manager*, that acts as a mediator to make sure that all components have a fair say.

We evaluated our approach in the context of pervasive workflow systems based on multiple simulations where we have compared a system with and without coordinated adaptation. Our results show that the convergence to a stable system is faster with the coordinated approach. For the future work we will investigate the following questions: how many proposals per round should be made,and how can the overall adaptation manager facilitate faster convergence to the overall solution.

## Acknowledgment

This work is partially funded by the FP7 EU FET project Allow IST-324449.

## References

[1] A. Soylu, P. D. Causmaecker, and P. Desmet, "Context and adaptivity in context-aware pervasive computing environments," *Ubiquitous, Autonomic and Trusted Computing, Symposia and Workshops on*, vol. 0, pp. 94–101, 2009.

[2] "ALLOW Project," http://www.allow-project.eu/.

[3] A. Bucchiarone, A. L. Lafuente, A. Marconi, and M. Pistore, "A formalisation of adaptable pervasive flows," in *WS-FM'09*, Bologna, Italy, 4 September 2009.

[4] K. Herrmann, K. Rothermel, G. Kortuem, and N. Dulay, "Adaptable Pervasive Flows - An Emerging Technology for Pervasive Adaptation," in *Workshop on Pervasive Adaptation (PerAda)*, September 2008.

[5] A. Marconi, M. Pistore, A. Sirbu, H. Eberle, F. Leymann, and T. Unger, "Enabling adaptation of pervasive flows: Built-in contextual adaptation," in *Proceedings of the 7th International Joint Conference on Service Oriented Computing (ICSOC'09)*, 2009.

[6] W. M. P. van der Aalst and K. M. van Hee, *Workflow Management: Models, Methods, and Systems*. MIT Press, 2002.

[7] S.-W. Cheng, A.-C. Huang, D. Garlan, B. Schmerl, and P. Steenkiste, "An architecture for coordinating multiple self-management systems," in *WICSA'04*, June 2004, pp. 243–252.

[8] V. Zamudio and V. Callaghan, "Unwanted periodic behaviour in pervasive computing environments," in *Pervasive Services, 2006 ACS/IEEE International Conference on*, June 2006, pp. 273–276.

[9] L. Capra, W. Emmerich and C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications," *IEEE Trans. Software Eng.*, vol. 29, no. 10, pp. 929–945, 2003.

[10] R. Kazhamiakin, M. Pistore, and A. Zengin, "Cross-layer adaptation and monitoring of service-based applications," in *MONA+*, 24 November 2009.

 ISBN: 978-1-61208-109-0 

# Towards a Resilient Message Oriented Middleware for Mission Critical Applications

Jinfu Wang[1], Beatriz Viñal Murciano[2], John Bigham[3]

MPI-QMUL ISRC[1]
Macao Polytechnic Institute & Queen Mary University of London
Macao SAR, China
jinfu.wang@mpi-qmul. org[1]

School of Electronic Engineering and Computer Science[1, 2, 3]
Queen Mary University of London
London, UK
john.bigham@eecs.qmul.ac.uk[3,]
beavimu@gmail.com[2]

***Abstract* — Message oriented middleware (MOM) provides a messaging service layer between the transport and application layer of the networking protocol stack. A resilient MOM system strives to provide a required level of message brokerage service in the face of bursty surges in workload demand, and failures in the underlay network or brokers. Resilience in our MOM system is achieved by a novel workload allocation mechanism which minimizes the quantified risk of workload exceeding capacity of a broker, while introducing redundant mirroring of workload; and also using resilient overlay routing and multi-homing to mitigate and recover from underlay network failure(s). This paper discusses the overall system architecture and the workload allocation and mirroring mechanism we employed. Comparing with round robin maximizing resource reserve ratio, our allocation algorithm provides superior resilience in minimizing the risk of correlated workload exceeding the capacity of system.**

***Keywords- message oriented middleware; resilience; self-adaptive; publish-subscribe; quantifying risk; self-healing; overlay routing.***

## I. INTRODUCTION

Mission critical applications, e.g., financial data delivery banking transactions, and remote control commands all place different requirements on publish subscribe message oriented middleware (PSMOM) for reliability and performance. They are very sensitive to performance degradation and network and broker failures. This paper looks particularly at the provision of resilience to broker and link failures and performance degradation, and also at the related topic of assured delivery.

PSMOM is software used to support communication between components both within a system and between cooperating systems [1][2]. It is especially suitable when a message needs to be distributed to multiple subscriber applications as it reduces the number of point-to-point connections required for applications to communicate. MOMs are compatible with, and indeed can form a central component of the Enterprise Service Bus (ESB) architecture [3], where the MOM message broker acts as the "bus" between applications. In practice because of issues relating to differing enterprise responsibilities, geographical locations, bottlenecks in performance [4], mitigation of single points of failure, multiple brokers will be deployed using both high-availability clustering and broker federation.

Resilience is the ability of the MOM system to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation, such as internal broker faults and external link failure and denial of service (DoS) attacks. To provide a resilient brokerage service, our PSMOM system is an overlay of federated brokers over a network, composed of several local domains, i.e., a few neighbour brokers instead of one centralized broker. The goal of the design is to provide adaptive configuration solutions and reactive mechanisms to provide a resilient service despite the risks from link or broker failures or degradation and surges in workload.

The resilience is achieved by: firstly, allocating the workloads to both primary brokers, and mirror brokers in the local domain - the mirror brokers provide redundancy at application end points in case of faults in primary broker; and secondly, employ overlay level multi-path routing in the overlay domain to provide reliable networking over wide area network between local domains in face of link faults and performance degradation. This paper discusses the first part, workload allocation and mirroring algorithm which quantifies and minimizes the risk of overloading brokers while exploring the correlations between messaging workloads; and also explains architecture and operation of the system.

This work has been driven by the requirements to produce resilient and secure MOM solutions appropriate for future real-time and business critical systems, which form the focus of the EU FP7 GEMOM project (Genetic Message-Oriented Secure Middleware) [5][13]. Besides the work that

 ISBN: 978-1-61208-109-0 

this paper is centered on, adaptive security and trust mechanism [7][8][11] are another strand of the GEMOM system. However, these will not be discussed in this paper.
The rest of the paper is structured as following. In Section II we present related researches and state of art MOM approaches. Then, in Section III the architecture and resilience mechanisms of our MOM system is explained. Section IV presents the risk-aware workload allocation and mirroring mechanisms, Section V explains the operation of system and Section VII describes an industrial application scenario. Finally, conclusions and ongoing works are briefly given in Section VI.

## II. BACKGROUND AND RELATED WORKS

Most state-of-the-art industrial MOM systems, e.g., Apache's AMQP Qpid, offer the high-availability clustering (HA clustering) and federation functionalities, to enhance reliability, interoperability and scalability of messaging service. The two mechanisms can be combined and adapted to build MOM deployments with different topologies and to suit scenarios where applications have different performance and reliability requirements. The current HA clustering techniques and federation techniques are orthogonal concepts. Broker clustering techniques consist in creating groups of brokers that work closely together. HA clusters improve the reliability by replicating entire states and messages of the working broker to another broker. They support the clients to failover to another broker in the cluster if the working fails. Federation enables the communication between brokers. Federation thus both supports connecting brokers in different domains as a MOM overlay, and improves system scalability by distributing computation and bandwidth contention of the message brokerage to multiple brokers such as [12].

Some problems associated with the current HA clustering and federation motivates our research. Firstly, the bursty surge in demand of workloads will cause significant performance degradation [15] and the surge of correlated workloads will have a supper added influence on such degradation. A workload allocation mechanism that minimizes such problem is missing. In our approach instead of only considering mean value of workloads, our system employs a novel workload allocation and mirroring algorithm which quantifies the probability of workloads exceeding capacity of brokers, while exploring the correlation between messaging workloads with variance and covariance matrix based method; Secondly, introducing redundancy with HA clustering requires replicating the entire workload of a primary broker to another broker, which requires much computation resource and the replication intra LAN does not fit well with MOMs deployed for internet messaging. With our algorithm, we mirror the partitions of workloads (called items) on a primary broker to different neighbor brokers in the local domain, instead of replicating entire its workloads to another broker. Thirdly, the federation of MOMs across internet requires low converge time from random underlay IP network failures and faults. Our system employs overlay level multi-path routing for resiliency in networking between local domains.

Resiliency in P/S based MOM is a popular research field, with some related works focusing on specific contexts. Yoon [17] designed a set of protocols to replace a faulty broker with an extra spare broker. The replacement is through recovering the connections to neighbor brokers by contacting an external directory service and recovering subscription tables from the reconnected neighbor brokers. Their protocol is for on-demand replication which means there is no live redundancy for continued messaging. The entire workload of faulty broker is replicated together similar as the failover in HA clustering. Our approach provides live mirroring for continued messaging in the face of single broker failure with workload mirroring and reconfiguration solution for multiple broker failures.
Kazemzadeh [6] describes an approach to handle broker failures, while maintains order and provides exactly once delivery of publications. Each broker maintains a partial topology mapping of brokers in its publication tree, and a faulty broker in the mapping is replaced by reconnecting to the next working broker in the mapping. The cost of their approach is to trade for stability with latency comparing with multi-path routing over internet. Their approach is designed to fit content based, not topic based P/S.

Finally some works, such as Snoeren et al. [16], employs overlay level multi-path routing or multi-homing, which has evolved from similar network level approaches [9][14], to build a fault tolerant P/S system, by constructing redundant disjoint forwarding paths between subscribers and publishers. While we employ similar overlay level multi-path routing for connecting disjoint domains with QoS awareness, we also improve the application end point resilience by workload allocation and mirroring in local domains as described in the following sections.

## III. OVERVIEW OF THE GEMOM SYSTEM ARCHITECTURE

The GEMOM resilience functions are designed to provide an acceptable level of continued messaging functionality in the face of faults or failures detected by the monitoring components. This is achieved by allocating primary workload and mirroring workload according to computed policies which minimize the risk of overloading brokers. As explained later the risk of system failure is the criterion that is used to assign load and mirrors. The overall traffic is allocated among a group of neighbor brokers intra cluster (e.g., in the case of Enterprise Messaging) or across internet (e.g., internet messaging). This group of neighbor brokers and the clients they serve is a local domain. Between disjoint local domains, overlay level multi-path routing can be employed to improve network link resilience, though this will not be discussed in the paper. This network model is shown in Fig. 1. The resilience architecture in GEMOM includes a Management Agent (MA) collocated

 ISBN: 978-1-61208-109-0 

with each broker. The Management Agent consists in components including Resilience Manager (RM), Overlay Manager (OM) and anomaly monitors.

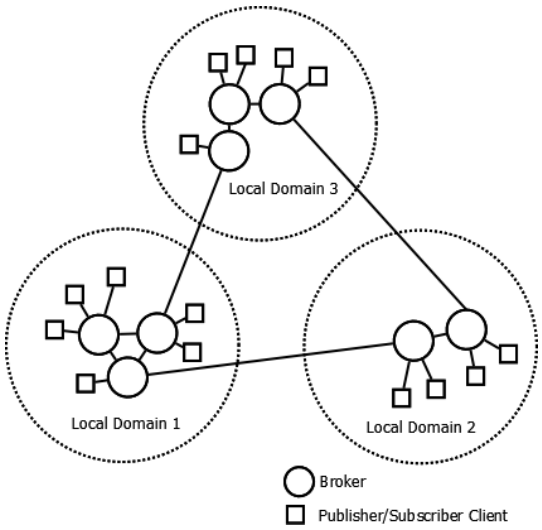



Figure 1. Network model of MOM system.

One of the MAs in the local domain is elected as Master MA, which is used to compute policies and distribute the policies and updates to other slave MAs. When a Master MA fails, one of the other MAs is elected as the new Master MA. The up-to-date policies in all MAs and the master election mechanisms provide resilience to master MA faults. Currently the host with most computation power is chosen as the Master MA, although different election algorithms can be applied. Inside the Master MA its RM and OM operate as following. RM computes the workload allocation policies and redundant workload replication policies which is called ***mirroring*** policies. These policies are computed for different possible state of the MOM system, e.g., initial allocation, single broker failure, double failure and RM builds a case base with these policies. Note that in low load situations the policies associated with switching off certain brokers are essentially the same as the failure case. The RM communicates with the OM and sends updates of case base of the new policy and its context. The OM is the management interface of to the broker and clients. One important functionality of OM is Broker Directory Service (BDS). Like DNS, BDS maps the topics to the physical brokers which act as their primary and mirror brokers. OM keeps BDS up to date according the policy case base and the current state of system. OM listens to the system state from monitoring tools. When a critical event, e.g., a broker failure, is under the radar, an optimal policy under current state of system retrieved from the case base is used by OM to update BDS. The Master MA distributes the changes in policy case base and BDS to all slave MAs to keep them up to date. These updates in BDS push brokers and clients to adapt to current policies.

The monitoring components consist of a set of anomaly detectors, namely broker failure, link failure and bottleneck detectors. The detected events from detectors which reside with each broker and each client trigger the adaptive actions in brokers and clients. The events are also passed to the Master MA to update current system policies.

A risk-aware workload allocation algorithm and hard constraints are used to compute the policies. Comparing with high-availability clustering, the policies are computed and applied on "item" level to reduce computation and the size of case base. Item is a system administrator defined partition of total workload carried by the MOM. An item can be considered as set of topics, treated by the RM as a single entity with respect to re-allocation of resources. A topic in a PS MOM is simply a label that a publisher (or publishers) can use to identify a message stream of particular type and subscribers can subscribe to the topics they wanted. Another difference is that a redundant replication, i.e., a mirror item, can be mirrored to live brokers instead of just to an idle slave broker to replicate entire workload and states of a live broker. Ideally this will allow a smaller set of brokers to carry the traffic at the prescribed risk bound. This algorithm is described in the following section. The overlay level-multipath routing between disjoint local domains is out the scope of this paper.

## IV. MESSAGING WORKLOAD ALLOCATION AND MIRRORING

Mission critical applications such as remote control commands and finical data are very sensitive to the service degradation and message loss due to link or broker failures. Service degradation is often caused by the workloads on a broker rise and exceed the processing capacity of the broker [15]. Correlated workloads often exist in real world systems. Such correlated workloads will likely rise together, and thus have a super additive effect on the total workload. Uncorrelated or negatively correlated workloads, on the other hand, will make the total workload more stable. In our MOM system the workloads in a local domain are allocated and mirrored among a few neighbor brokers. RM employs an workload allocation and mirroring algorithm to quantify and minimize the risk of workload exceeding a broker while explore and mitigate such correlation. RM computes workload allocation policies and redundant workload replication policies, i.e., *mirroring* policies, under different possible states of the MOM system, e.g., initial allocation, single broker failure, double failures, using this algorithm. The policies are stored in a case base and retrieved according to current state of system as described in the previous section.

In comparison to HA clustering, where the entire workloads of a primary broker is replicated to another live broker, our mirroring strategy divides and assigns the mirror workloads of a primary broker to several live neighbor brokers in local domains, and does not necessarily introduce

 ISBN: 978-1-61208-109-0 

many extra idle brokers. The atomic unit for the workload allocation and mirroring is an *item*. An item, according the specific scenario where the MOM system is deployed, is a pre-defined partition from the total traffic carried in MOM system. For example, in the case of topic-based PS MOM, an item is defined as one or more topics that are bundled together in workload allocation. Every item is mapped to a primary and a mirror broker. A physical broker logically can be the primary broker for some and the mirror broker for some other items at the same time.

The RM computes workload allocation and mirroring policies to minimize the quantified risk of overload to the system, and also subject to satisfying important prescribed hard constraints such as requiring that the probability the workload exceeds each broker's capacity is lower than a prescribed upper bound or that the maximum number of topics that a broker is allowed to carry is not exceeded. This problem is formulated as allocating a set of items (an item is also a set, but is a set of topics) *j* to each broker *i*. Consider the optimal primary allocation case, i.e., the case where we simply want to maximize the utility function while allocating items to the brokers which are candidates that satisfy the pre-defined constraints. Then the goal is to find the optimal solution $S = \{j_1, \dots j_k\}$, which is k sets of items allocated to the corresponding k brokers, maximizing overall system utility, i.e., minimizing the overall risk, in the system. The peak period message rate of item set j is a random variable denoted by $\boldsymbol{V^j}$ and the capacity of broker *i* is a constant $C_i$ (both measured in messages per unit time). Given the utility of current local domain $U_{domain}$, the problem is:

$$\arg\max_{S=\{j_i\}} (U_{domain} = \sum_{i=1}^{k} U_i) \qquad (1)$$
$$where\ U_i = R_i^j - \int_{C_i}^{\infty} P_i^j (x - C_i) P_r(V^j = x) dx$$

The utility $U_i$ of broker i carrying item set j is calculated from $R_i^j$ which is the reward for carrying item set j and the penalty $P_i^j(x - C_i)$ for the arriving messages x exceeding the capacity of broker i. $R_i^j$ is a value associated with item set j and $P_i^j$ ideally is proportional to $(x - C_i)$.
By assuming $V^j$ follows a normal distribution and relaxing $P_i^j$ to a value associated with item set j, we simplify (1) as:

$$\arg\max_{S=\{j_i\}} (U_{domain} = \sum_{i=1}^{k} U_i) \qquad (2)$$
$$where\ U_i = R_i^j - P_i^j P_r(V^j > C_i)$$

We can approximate by $P_r(V^j > C_i) = P_r(\frac{V^j - \mu_j}{\sigma_j} > \frac{C_i - \mu_j}{\sigma_j})$. $\frac{V^j - \mu_j}{\sigma_j}$ follows standard normal distribution, and the parameters are estimated by analyzing peak period message rates samples of all topics. The message rate sample is a series of messages arriving at unit time for each topic in the system over peak periods. We can have mean message rate $\mu_t$of each topics t. Given $V^j = \sum_{t \in j} V_t, \mu_j = \sum_{t \in j} \mu_t$. From the sample series of all topics T, we calculate a variance covariance matrix of $T \times T$. $\sigma^2 = \sum_{a \in j} \sum_{a' \in j} cov(V^a, V^{a'})$ where a and a' are any item in j; and $cov(V^a, V^{a'}) = cov(\sum_{t \in a} V^t \sum_{t' \in a'} V^{t'})$, where t and t' are any topic in a and a'. Intuitively covariance measures the degree to which two variables change or vary together, i.e., co-vary. The positively correlated items' message rates vary together in the same direction relative to their expected values, hence they result in a relative larger $\sigma$, which leads to a larger $P_r(V^j > C_i)$ *and smaller* $U_i$. Hence by maximizing $\sum_{i=1}^{k} U_i$) the allocation solution will result in a small system risk of being overloaded and positively correlated items are less likely to be allocated to the same broker.

To illustrate the allocation algorithm, there are overall 32 topics in a local domain, which are divided into 6 items, to be allocated to 3 brokers with different capacities in table 3. From sample series data, the mean and the variance covariance matrix among 8 items are calculated as in table 1 and 2. We find the solution for (2) with a Depth First Search using $R_i^j = P^j = \mu_j$ as the reward and penalty associated with j. We apply a prescribed risk threshold=0.007 as a hard constraint (3) to the search:

$$s.t.\ P_r(V^j > C_i) < threshold \qquad (3)$$

We compare the allocation result with a round robin allocation which allocates each item in turn to a broker with maximum resource reserve ratio (4). The two solutions are shown in table 3.

$$reserve\ ratio = \frac{C_i - V^{j'}}{C_i} \qquad (4)$$

The risks of two solutions are evaluated by the normalized gain (5). Risk in our solution and round robin solution is $\mathbf{Pr}_1$ ***and*** $\mathbf{Pr}_2$. A postive gain indicates $\mathbf{Pr}_1 < \mathbf{Pr}_2$. The gain is shown in Fig.2.

$$Gain = \frac{(\Pr_2(V_i^{j'} > C_i) - \Pr_1(V_i^j > C_i))}{\Pr_2(V_i^{j'} > C_i)} \qquad (5)$$

In this illustration, in term of the estimated risk our solution shows obvious advantage over round robin solution. It is because our allocation algorithm avoids allocating some highly correlated workloads to the same broker by exploring the correlation via the variance computed from the variance covariance matrix.

 ISBN: 978-1-61208-109-0 

TABLE I. MEAN MESSAGE RATES OVER SAMPLED PERIODS

| Item1 | Item2 | Item3 | Item4 | Item5 | Item6 |
|---|---|---|---|---|---|
| 150 | 300 | 350 | 200 | 100 | 130 |

TABLE II. VARIANCE COVARIANCE MATRIX OF 8 ITEMS

| | Item1 | Item2 | Item3 | Item4 | Item5 | Item6 |
|---|---|---|---|---|---|---|
| Item1 | 5310 | 2456 | 962 | 1071 | 840 | 409 |
| Item2 | 2456 | 7544 | 611 | 418 | 1237 | 542 |
| Item3 | 962 | 611 | 6972 | -1622 | 3401 | 2387 |
| Item4 | 1071 | 418 | -1622 | 7538 | -102 | 4266 |
| Item5 | 840 | 1237 | 3401 | -102 | 4992 | 3626 |
| Item6 | 409 | 542 | 2387 | 4266 | 3626 | 4202 |

TABLE III. THE CAPACITY AND SOLUTION OF EACH BROKER

| | Broker 1 | Broker 2 | Broker 3 |
|---|---|---|---|
| $C_i$ | 700 | 750 | 600 |
| Solution 1 | Item 2,6 | Item 1,3 | Item 4,5 |
| Solution 2 | Item 1,2 | Item 3,5 | Item 4,6 |
| $P_{r_1}(V^j > C_i)$ | 0.0018 | 0.0016 | 0.0010 |
| $P_{r_2}(V^{j'} > C_i)$ | 0.0068 | 0.0054 | 0.0170 |

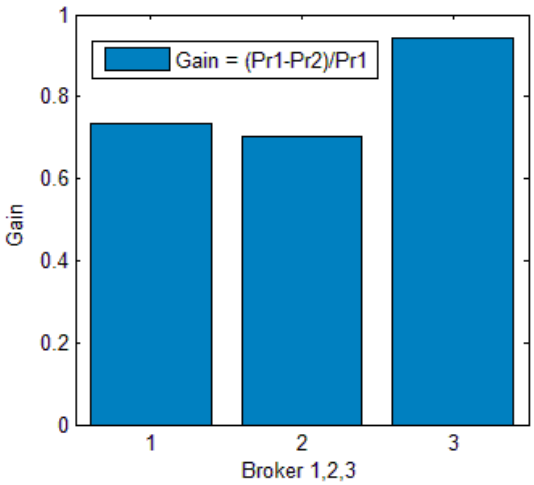



Figure 2. Normalized gain of our solution over round robin at each broker.

## V. SYSTEM OPERATION

In each local domain, the Master MA computes the policy case base, and updates BDS in according to the current optimal policy as described in section III. The BDS entries is a 3-tuples in the form of [topic-name@domain, primary-broker, mirror-broker]. The BDS entries and are replicated to slave MAs and pushed to publishing/subscribing clients. Each client connects to both primary and mirror brokers according to the up-to-date BDS entries. The mirroring approach intrinsically trades off redundant resource to reliable messaging service. According to the QoS requirement of different topics, the critical topics with higher requirements for reliability and speed are published simultaneously to both primary and mirror broker; while the topics with lower requirements are published only to primary broker, however a session is pre-established to the mirror broker – the topic starts to publish to mirror broker at the moment when faults are detected in primary broker. Each message published by the same client (including the same message published through different paths) has an unique ID, in one overlay node, only at most one copy of message can exist. The local domain and inter domain messaging is illustrate in Fig. 3.

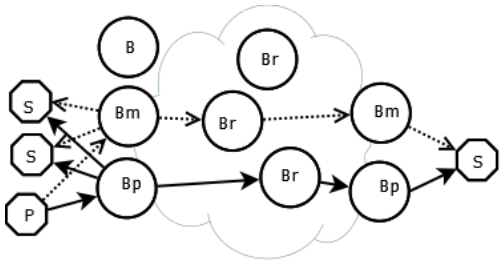



Figure 3. The operation of overall system.

In Fig. 3, primary and mirror brokers (Bm and Bp) provides the clients with both local domain redundancy and overlay level disjoint paths to remote domains.

## VI. DEPLOYMENT SCENARIO

GEMOM is designed to emphasis on enterprise messaging where a few independent local domains which consist of a few neighbor brokers, are interconnected as the overlay domain by federation over WAN. For example in one of the case studies, a MOM overlay is deployed to provide messaging service for a national high way monitor and control system. Traffic sensors, mobile clients, toll booth, control centers and other components exchange messages via the MOM overlay. The local domain brokers are distributed across the different parts of the nation. The mirroring approach allows continued message delivery at the application end points in the case where primary broker(s) fail, whether from malfunction or as a consequence of degradation for a DoS attack, or indeed from simply switched off for maintenance. Although disjointedness of underlay network paths, e.g., not sharing a common border gateway, cannot be assured since we do not assume that we have no control over the substrate ISP network, the disjoint placement of brokers carrying primary and mirror workload, and overlay level multi-path routing between federated brokers improve resilience of messaging service in the face of networking faults and failures [9][14].

 ISBN: 978-1-61208-109-0 

## VII. CONCLUSION AND FUTURE WORK

In this paper, we presented the architecture and operation of the GEMOM system and the risk-aware workload allocation and mirroring mechanisms employed. The allocation algorithm minimizes the risk in a quantified way and mitigates correlated bursty workload to different brokers. The mirroring solution adapts the system to tolerate broker and faults and failures in the application end points in local domains. These mechanisms together with overlay level multi-homing and routing constitute the GEMOM approach towards a resilience MOM system.

In future work, we will develop a more accurate model that profiles broker capacity, and novel scheduling mechanisms are being researched to support QoS awareness and improve performance of overlay level multi-path routing connecting different domains.

## ACKNOWLEDGMENT

This research is in the context of the EU project GEMOM grant agreement no 215327, approved by the EU Commission and TSB REMOM project. The authors acknowledge the efforts of all partners (Q-Sphere, NR, VTT, TXT, CNIT, JRC, Datel) who participated in the development of the GEMOM framework. The first author had been partly supported by Macao STDF funding.

## REFERENCES

[1] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec, "The Many Faces of Publish/Subscribe," Computing Surveys, vol 35, PART 2. USA, ACM, 2003, pp. 114-131.

[2] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White, "Towards Expressive Publish/Subscribe Systems," Lecture Notes in Computer Science, NUMB 3896, Germany, Springer-Verlag, 2006, pp. 627-644.

[3] ESB. Enterprise Service Bus (ESB). Accessed on Sept. 8th, 2010, from http://www.progress.com/psm/sonic/enterprise-service-bus/index.ssp.

[4] P. Tran, P. Greenfield, and I. Gorton, "Behavior and Performance of Message-Oriented Middleware Systems," Proc 22nd International Conf. on Distributed Computing Systems Workshops, IEEE press, 2002, pp. 645 – 650.

[5] J. Wang, J. Peng, J. Bigham, B. Chew, B. V. Murciano, M. Novkovic, and I. Dattani, "Adding Resilience to Message Oriented Middleware," Proc. 2nd International Workshop on Software Engenering for Resilient Systems (SERENE 2010), UK ACM Press, 2010 pp. 89-94.

[6] Y. Yoon, V. Muthusamy, and H.-A. Jacobsen "On-demand Replication for Failover in Content-based Publish/Subscribe Overlays," Middleware Systems Reserch Group Technical Report, University of Toronto, November, 2009.

[7] H. Abie,, R. Savola, and I. Dattani, "Robust, Secure, Self-Adaptive and Resilient Messaging Middleware for Business Critical Systems," 1st International Conference on Adaptive and Self-adaptive Systems and Applications, Athens. IEEE Press, Nov. 2009, pp. 153 – 160.

[8] H. Abie, I. Dattani, M. Novkovic, J. Bigham, S. Topham, and R. Savola, "GEMOM - Significant and measurable progress beyond the state of the art," Proc 3rd International Conference on Systems and Networks Communications (ICSNC 2008), IEEE Press October, 2008. pp. 191-196.

[9] V. Vasudevan, D. G.Andersen, and H. Zhang; "Understanding the AS-level Path Disjointness Provided by Multi-homing."; Technical Report RCMU-CS-07-141, School of Computer Science, Carnegie Mellon University

[10] J. Mirkovic and P. Reiher, "A Taxonomy of DDOS Attacks and DDOS Defense Mechanisms," Computer Communication Review 2004, Vol 34, Numb 2. ACM USA ISSN 0146-4833, pp. 39-54.

[11] J. Wang, and Bigham, J. 2008. "Anomaly detection in the case of message oriented middleware," Proc 1st International Workshop on Middleware Security (MidSec '08), ACM, New York, 2008, pp. 40-42.

[12] G. Marsh, A.P. Sampat, S. Potluri, and D.K. Panda, "Scaling Advanced Message Queuing Protocol (AMQP) Architecture with Broker Federation and InfiniBand," technical report, available at ftp://ftp.cse.ohio-state.edu/pub/tech-report/2009/TR17.pdf, accessed on Sept. 8th, 2010.

[13] H. Abie, R. Savola, J. Wang, and D. Rotondi, "Advances in Adaptive Secure Message Oriented Middleware for Distributed Business Critical Systems," To Appear on Proc 8th International Conference of Numerical Analysis and Applied Mathematics (ICNAAM 2010), Rhodes, Greece, September 2010

[14] A. Akella, J. Pang, B. Maggs, and S. Seshan, "A comparison of overlay routing and multihoming route control," Computer Communication Review, 2004, ACM Press, VOL 34 PART 4, Pp. 93-106.

[15] N. Mi, G. Casale, L. Cherkasova, and E. Smirni, "Burstiness in multi-tier applications: Symptoms, causes, and new models," Proc 9th ACM/IFIP/USENIX International Conference on Middleware (Middleware 2008), Springer-Verlag, New York, 2008, pp. 265-286.

[16] A. C. Snoeren K. Conley, D. K. Gifford, "Mesh-based content routing using XML," in ACM SIGOPS Operating Systems Review Volume 35 , Issue 5, ACM Press, 2001. pp: 160 - 173

[17] R. Kazemzadeh and H.-A. Jacobsen "Reliable and Highly Available Distributed Publish/Subscribe Service," Proc 28th IEEE International Symposium on Reliable Distributed Systems, IEEE Press, 2009, pp. 41-50.

 ISBN: 978-1-61208-109-0 

# Hidden State Observation for Adaptive Process Controls

Melanie Senn, Norbert Link
*Institute of Computational Engineering at IAF*
*Karlsruhe University of Applied Sciences*
*Moltkestrasse 30, Karlsruhe, Germany*
*Email: melanie.senn@hs-karlsruhe.de, norbert.link@hs-karlsruhe.de*

***Abstract*—In many manufacturing processes it is not possible to measure on-line the state variable values that describe the system state and are essential for process control. Instead, only quantities related to the state variables can be observed. Machine learning approaches are applied to model the relation between observed quantities and state variables. The characterization of a process by its state variables at any point in time can then be used to adequately adjust the process parameters to obtain a desired final state. Also, multiple process controls of a process chain can be linked using standardized transfer state variables between the single processes. This allows the optimization of the entire process chain with respect to the desired properties of the final workpiece. This paper proposes a general method to extract state variables from observable quantities by modeling their relation from experimental data with data mining methods. After transforming the data to a space of de-correlated variables, the relation is estimated via regression methods. Using Principal Component Analysis and Artificial Neural Networks we obtain a system capable of estimating the process state in real time. The feasibility of our approach is shown with data from numerical simulation of a deep drawing process.**

***Keywords*-statistical process model; hidden state prediction; regression analysis; dimension reduction; deep drawing.**

## I. INTRODUCTION

Closed-loop controls are capable of reaching desired final states by compensating disturbances in single processes or by adapting to varying input in a process chain. Feedback about the system state is essential for this purpose. The measurement of the real state variables usually requires large efforts and cannot be executed in process real time. Only few process-related quantities can be measured by real production machines during process execution. If these observables can be related to state variables with sufficient unambiguity and accuracy, a state-based closed-loop control can be created. The final state can then be estimated as well and the information be transferred to the control of the next step in a process chain. In deep drawing, observables such as forces and displacements in the tools and the workpiece are accessible during process execution with reasonable measurement effort. Mechanical stress distributions reflecting the state of the sheet material can be used as controlled variable as applied in [1] to optimize the blank holder force for an experimental deep drawing environment. A control system for deep drawing is presented in [2], based on the identification of static material properties as proposed in [3].

Data mining methods for regression analysis such as Artificial Neural Networks (ANNs) or Support Vector Regression (SVR) are widely used in material science for the prediction of static process quantities. In [4], thickness strains have been computed, [3] presents a model to predict material properties from process parameters and conditions. These both affect the final result, however, conditions are constant during execution and can not be used for on-line state control. The texture of cold rolled steels has been predicted from process conditions in [5]. A general overview for the application of ANNs in material science is given in [6] under consideration of model uncertainties and noise.

In our approach a feedforward, completely connected ANN is used due to its capability of modeling the nonlinear relation between observable quantities and the process state. Principal Component Analysis (PCA) is applied for dimension reduction in observables and state variables to decrease the complexity of their relations.

This paper is structured as follows. In Section II the statistical process model and underlying data mining methods are introduced. A proof of concept is given by the application of the statistical model to data from numerical experiments of a deep drawing process in Section III. Results for predicted state quantities are presented and evaluated in Section IV. Section V concludes and outlines future work.

## II. MODELING

Numerical models based on first principles have the ability to predict results accurately and reliably after they have been validated by experimental results. However, the high quality comes along with high computational costs. Phenomenological models are based on observations of first principles and normally require less, but still substantial computational resources. Both types of models can be used to describe dynamic process behavior. If it comes to on-line process control, however, high speed models are needed to make fast predictions. Statistical models provide this property and thus can be used to reproduce the relation between observable quantities and process states on the one hand and the relation between state variables and appropriate process parameters on the other hand.

 ISBN: 978-1-61208-109-0 

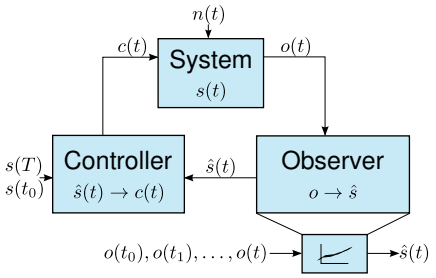



Figure 1. Closed-loop adaptive process control

### A. Relating Observables to State Variables

During process execution, the dynamic system moves along in its state space where each state generates observable values related to the respective state variable values. In materials processing, the state variables may be fields of intensive magnitudes such as strains or stresses that are reflected in observables like displacements, forces and temperatures.

A closed-loop adaptive process control based on hidden state observation is shown in Figure 1. The dynamic system is characterized by its state $s(t)$ and is subject to a system noise $n(t)$ that has to be compensated by the controller to reach a defined final state $s(T)$. The observer models the relation between observables and state variables and delivers estimated state variables $\hat{s}(t)$. These are again used by the controller to find appropriate process parameters $c(t)$ considering the reference $s(T)$ as definition for the final state at time $T$. If multiple process controls are linked to a process chain, the final state of the preceding process serves as initial state of the current process $s(t_0)$. This additionally influences the process parameters determined by the controller during process execution.

The hidden state observer provides state information by deriving the current estimated state $\hat{s}(t)$ at time $t$ from observables between process begin at time $t_0$ and the current time $t$. Input and output quantities of the regression analysis are high dimensional, whereas with reasonable measurement effort only a limited number of samples can be provided by experiments. Therefore, we propose to model the complex relation between observables and state variables with an ANN applying dimension reduction to input and output before regression analysis is performed.

### B. Regression Analysis

A feedforward, completely connected ANN is used to model the nonlinear relation between observables (input) and state variables (output). We choose a three layer network topology (input, hidden, output), which is sufficient according to the theorem of Kolmogorov [7]. Each of the neurons in the subsequent layer is connected to all neurons of the current layer, where each connection has assigned a certain weight value. A logistic activation function is applied to the superposition of the activations of preceding neurons and the weights added up with a threshold value. The regression analysis by means of ANNs consists of minimizing an error cost function with respect to the weights and thresholds. For the cost function, the sum of squared errors (SSE) between the output values of the network and the output values of the associated input values as given by a sample is selected. The ANN is trained by the backpropagation algorithm [8].

The number of nodes in the hidden layer has been determined according to (1), see [9]. The objective is to retrieve an overdetermined approximation, i.e., the number of training samples must be greater than the number of degrees of freedom, namely the number of connection weights.

$$KN = \alpha(J(I + K) + J + K) \tag{1}$$

Equation (1) reveals the relation between

- the number of input nodes ($I$)
- the number of hidden nodes ($J$)
- the number of output nodes ($K$)
- the number of training samples ($N$)
- the determinacy factor ($\alpha$),

which is problem dependent. Starting from a minimum of 1.0 (exact determination), the optimal determinacy factor $\alpha$ has been experimentally identified by evaluation of the network's performance function quantified by the mean squared error (MSE). A first guess of determinacy has been obtained by comparison of network performance results between 1.0 and the maximum determinacy resulting from a network with only one output node with a step width of 10. Successive refinements by step widths of 1 and 0.1 have been performed around the value of the previous step until the optimal determinacy is obtained with respect to the network's performance function.

The number of output nodes is on the one hand predefined by the number of output dimensions of the regression problem itself, but on the other hand the output nodes do not necessarily have to belong to one single network. An extreme configuration is to generate one network per output dimension to reduce the complexity that has to be described by the hidden layer. In our approach we use only one network since the complexity of the regression problem has already been reduced by dimension reduction. The Levenberg-Marquardt algorithm is used to solve the optimization problem of finding optimal connection weights.

### C. Dimension Reduction

PCA is employed to reduce the dimensionality of observables and state variables by removing correlations in space and time. Before applying the PCA algorithm, the data spanned by the three dimensions

- the number of samples ($I$)
- the number of variables per time frame ($J$)
- the number of time frames ($K$)

have to be arranged in two dimensions. Reference [10] states that only two of the six possible unfolding techniques have

 ISBN: 978-1-61208-109-0 

practical relevance. In $A$-unfolding ($KI \times J$) the number of time frames and the number of samples are aggregated in the first dimension, the number of variables per time frame characterizes the second dimension. $D$-unfolding ($I \times KJ$) uses the number of samples as first dimension and combines the number of time frames and the number of variables per time frame in the second dimension. The latter is therefore more appropriate to remove correlations between different time frames as well as between variables within the same time frame. In [11], dynamic process behavior is monitored by Dynamic Principal Component Analysis (DPCA) considering a limited window of time lagged observations. In our model we use the full history of observables to make use of the complete information available to us.

The data in original dimensions $X$ are subject to a transformation of the principal axes by finding directions of maximum variance. The first new axis points in the direction of largest variance of the data $X$ and is called the first principal component. The second principal component is orthogonal to the first one and points in the direction of second largest variance. Additional components can be found analogously, while higher ones describe less variance.

$$X = \sum_{i=1}^{n} \alpha_i e_i \tag{2}$$

The data can be represented by (2), where $n$ stands for the number of dimensions of $X$, $e$ represents the basis vectors and $\alpha$ describes the data in the new coordinate system. Dimension reduction can be achieved by removing higher principal components since they do not explain much of the variance in the data.

$$K = \frac{1}{n-1} X^T X \tag{3}$$

Related eigenvectors and eigenvalues can be calculated from the covariance matrix $K$, see (3), where $X$ has been mean-centered before. Pairs of eigenvalues and eigenvectors are then sorted such that the largest eigenvalue is associated with the first principal component explaining most variance [12]. The covariance matrix can be seen as a description of the rotation in the transformation of the principal axes, the data centroid corresponds to the displacement of the origin of the new coordinate system with respect to the initial one.

If the number of variables is much greater than the number of samples, which might apply to observables, [13] advises to use Singular Value Decomposition (SVD) according to (4) to determine eigenvalues and eigenvectors efficiently.

$$X = USV^T \tag{4}$$

The eigenvalues $\alpha$ can be extracted from the diagonal matrix $S$ by (5) where $m$ corresponds to the number of samples, while the orthonormal matrix $V$ contains associated eigenvectors $e$.

$$\alpha = \frac{1}{m-1} S^T S \tag{5}$$

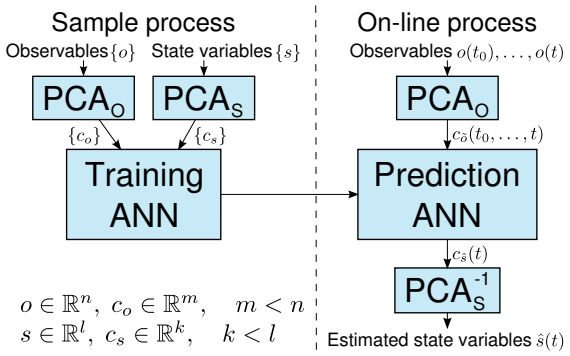



Figure 2. Architecture of the statistical process model

### D. Statistical Process Model

The statistical process model for hidden state observation displayed in Figure 2 is divided into a training and a prediction block. For each requested point in time $t$ the system collects previously sampled observables $o(t_0), \ldots, o(t)$ and current state variables $s(t)$. First, PCA is applied to both observables $o$ and state variables $s$ of which a subset is used to train the ANN as input $c_o$ and target $c_s$, respectively. After successful training the ANN is able to predict state variables in reduced dimensions $c_{\hat{s}}(t)$ from previously unseen observables $o(t_0), \ldots, o(t)$, reduced to $c_{\tilde{o}}(t_0, \ldots, t)$, that have not been included in training. The predicted state quantities $c_{\hat{s}}(t)$ are subject to an inverse dimension transformation to obtain their counterparts $\hat{s}(t)$ in the original, high dimensional space for visualization and validation.

## III. Application to Deep Drawing

The feasibility of the proposed approach is tested with an elementary sample process, the cup deep drawing of a metal sheet. In cup deep drawing, a metal sheet is clamped between a die and a blank holder. A punch presses the sheet that undergoes a traction-compression transformation into the die opening to obtain a cup-shaped workpiece. The assembly is displayed in Figure 3. Statistical samples have been generated by experiments performed in a numerical simulation environment. Fur this purpose an axisymmetric finite element deep drawing model has been implemented in ABAQUS (finite element analysis software).

Observable quantities are displacements and forces, temperature behavior has been neglected. Displacements in the cup bottom in direction of the moving punch have been recorded as well as displacements in the sheet edge in orthogonal direction to reflect sheet retraction. Additional displacements in punch and blank holder have been acquired. Reaction forces in the tools have been recorded in both radial and axial direction. Arising partial correlations in observables are removed by the PCA. The state of the deep drawing process is characterized by the von Mises stress distribution within the workpiece.

 ISBN: 978-1-61208-109-0 

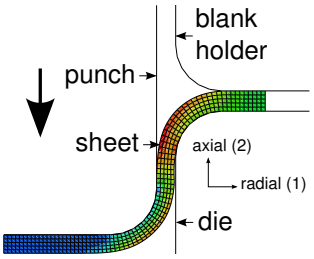



Figure 3. Workpiece and tools in axisymmetric deep drawing

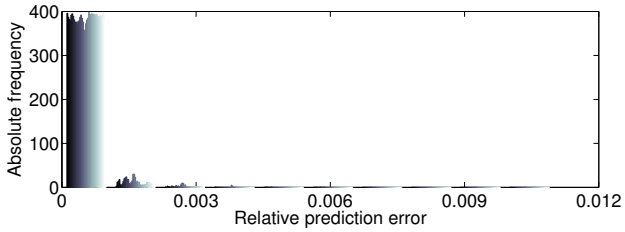



Figure 4. Relative error distribution for predicted state variables

In the performed parametric study, the blank holder force has been varied in the range of [70, 100] KN, process conditions such as drawing height or lubrication have been kept constant. For each sample observables and state variables have been collected for all time frames. A time frame equalization ensures common time frames for all samples. 200 samples have been generated, each consisting of 131 time frames, which in turn contain nine observables and 400 state variables. The extracted data have been randomly partitioned into a training set (80%) and a test set (20%). Dimension reduction has been applied to all samples, where the training data were split again randomly into a training set (80%) and a validation set (20%) for regression analysis. The test set has been used for overall validation of the statistical model. Resampling and subsequent remodeling has been performed to select the best model and to prove independence of specific data sets.

## IV. Discussion of the Results

Two use cases were identified for state estimation. The first use case refers to the prediction of the final process state based on the observable values during the entire process execution. This provides a subsequent process with detailed information about its input, allowing it to optimally adjust its parameters. The single process controls of a process chain can be linked by the state information in a standardized way, resulting in an overall quality improvement. This use case is described in Section IV-A. Prediction of the state evolution during process execution is employed in process control as discussed in Section IV-B. The latter can be seen as a generalization of the former.

### A. Prediction of the Final Process State

The statistical model for hidden state observation has been validated by a test set of 40 samples (see Section III). The relative prediction error of the 400 state variables never exceeds 0.0110 for all samples, the resulting distribution is shown in Figure 4. The absolute frequency of the number of state variables is high for small errors and drops rapidly with increasing error. Different colors stand for individual samples. The quality of the results shows the principle feasibility of the method. One must be aware that this might be partly due to the simplicity of the experiments: the variance in observables and state variables is not very large since the blank holder force has been the only varied parameter. Investigations with more realistic process models and real experimental data are subject of ongoing work.

The prediction quality was further analyzed as follows. It has been shown that the variation of predicted results is substantially smaller than the variance of the generated data. For this, we have defined the model uncertainty $\sigma$ by

$$\sigma = \frac{1}{n} \sum_{i=1}^{n} \frac{MSE_i}{Var_i}, \tag{6}$$

which corresponds to the mean value over all dimensions $i$ of the $MSE$ over all samples divided by the corresponding sample variance $Var$. In (6), $n$ stands for the number of dimensions, i.e., the number of state variables. A low model uncertainty is characterized by a $\sigma$ value close to zero, whereas values approaching 1.0 indicate high uncertainty. For our experiment $\sigma$ is 0.0045, which indicates the high accuracy and low uncertainty of the predicted results.

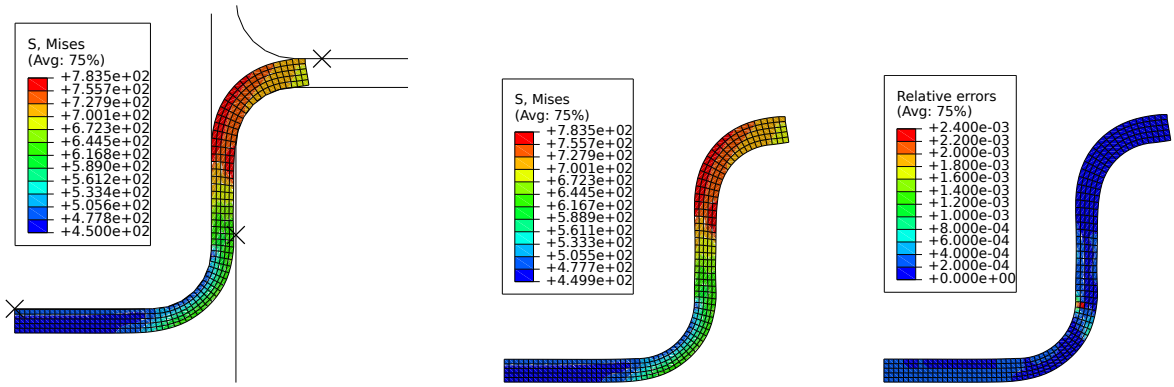
The quality of the statistical model has been quantified by the coefficient of determination $R^2$, which can be applied to nonlinear regression analysis by (7) according to [14]

$$R^2 = 1 - \frac{SSE}{SST}. \tag{7}$$

The $SSE$ describes the sum of the squared deviations between original data in the test set and associated predicted results. It is divided by the $SST$, which quantifies the variation in the test set calculated by summed squared deviations of the original state variables from their means. Both quantities are computed over all state variables for all samples in the test set. A resulting $R^2$ value of 0.9991 confirms the good quality of the statistical model. The MSE of the predicted state variables serves as a base for the determination of a confidence interval for the prediction error. The precision of the estimation amounts to a mean value of 0.0440, which has been calculated over all predicted state variables for a 95% confidence interval.

The overall error of the statistical model is composed of a time frame equalization error, the error from dimension reduction and the ANN prediction error. The MSE of the ANN amounts to 0.0019 at a typical range of [400, 800] MPa of the predicted von Mises stresses. Observables have been reduced from 1179 (9 observables per time frame $\times$ 131 time

 ISBN: 978-1-61208-109-0 




(a) Finite element model (original) (b) Statistical model (prediction) (c) Relative prediction errors

Figure 5. Comparison of results of the finite element model and the statistical model

frames) to nine dimensions with a predefined precision of 99.999% and thus a relative error of 0.001%. State variables have been reduced from 400 to seven dimensions with a precision of 99.900%, i.e., a relative error of 0.1%. On the one hand dimension reduction implies information loss that cannot be recovered, but on the other hand it enables the ANN to find correlations in the reduced data. A worse result might have been obtained without the usage of dimension reduction due to the huge number of additional degrees of freedom to be determined by the ANN.

Some results for a representative of the test set visualized in ABAQUS are depicted in Figure 5. It displays the absolute von Mises stress values in MegaPascal predicted by the statistical model in Figure 5b, which are in very good agreement with the results of the finite element model illustrated in Figure 5a. To outline the deviation of the predicted results from the original data the relative error in the range of [0, 0.0024] is presented in Figure 5c. Errors are low in regions with small sheet deformations, while higher errors occur in areas with high deformation gradients.

Robust predictions are characterized by bounded prediction errors despite of model uncertainties and disturbances. In our work we first applied a white noise of 5% to the observables to model a measurement error. State variables have then been predicted with a relative error in the range of [0.0, 0.0569] and a corresponding mean value of 0.0024. The model uncertainty $\sigma$ amounts to 0.1069, while the model quality is characterized by a $R^2$ value of 0.9313. Increasing the noise to 10% results in a relative error range of [0.0, 0.1115] with a mean of 0.0033, a model uncertainty $\sigma$ of 0.1990 and a $R^2$ value of 0.8412. The size of the error range does not solely represent the quality of prediction, also the model uncertainty affecting the distribution within this range has to be considered. The results indicate that our model is robust to small disturbances and still delivers satisfactory results for small manipulations in observables. However, with increasing noise model quality decreases as uncertainty increases.

### B. State Prediction During Process Execution

Process execution time determines the timespan in which process parameters can be adjusted to control the process state. State information is not necessarily needed for each single time frame, since controllers are usually liable to a certain delay in their impact. The implementation of the statistical model offers the selection of time frames that are crucial for control. In this work some representatives have been chosen to demonstrate the feasibility of state evolution prediction. Time frame numbers 1, 45, 90 and 131 have been selected, the results are outlined in Table I.

The parameters of the statistical process model have been set as follows. The respective determinacy of the ANN has been identified incrementally by evaluating the network's performance function (see Section II-B), the precision for dimension reduction has been chosen as 99.999% for observables and 99.900% for state variables. The number of observables and state variables in reduced dimensions each grows with increasing time since their inner relations become more complex. The number of hidden nodes increases as well due to the more complex relation between observables and state variables. Between time frame number 45 and 90 the number of hidden nodes however decreases. At this point the number of input nodes of the ANN given

Table I
PREDICTION CHARACTERISTICS DURING PROCESS EXECUTION

| Time frame number | 1 | 45 | 90 | 131 |
|---|---|---|---|---|
| # PCA observables | 1 | 1 | 6 | 9 |
| # PCA state variables | 1 | 2 | 3 | 7 |
| # ANN hidden nodes | 33 | 36 | 26 | 38 |
| ANN determinacy | 1.3 | 1.8 | 1.5 | 1.4 |
| MSE PCA observables | 1.2453 | 1.3612 | 51.8063 | 80.0205 |
| MSE PCA state variables | $4 \cdot 10^{-5}$ | $4 \cdot 10^{-4}$ | 0.0026 | 0.0667 |
| MSE ANN | $2 \cdot 10^{-5}$ | $7 \cdot 10^{-6}$ | $3 \cdot 10^{-4}$ | 0.0019 |
| $R^2$ statistical model | 0.9999 | 0.9999 | 0.9998 | 0.9991 |
| $\sigma$ statistical model | 0.1452 | 0.0003 | 0.0028 | 0.0045 |
| Mean relative error | 0.0047 | $8 \cdot 10^{-6}$ | $3 \cdot 10^{-5}$ | $2 \cdot 10^{-4}$ |
| Max relative error | 0.1071 | 0.0021 | 0.0020 | 0.0110 |

 ISBN: 978-1-61208-109-0 

by the number of observables in reduced dimensions is for the first time higher than the number of output nodes given by the number of state variables in reduced dimensions. The MSE caused by dimension reduction in observables and state variables rises with increasing time and thus increasing complexity. The performance of the ANN evaluated by its MSE decreases between time frame number 1 and 45 and then increases. This behavior is also reflected in the relative error distribution specified by its mean and maximum value. The explanation is composed of two opposed effects. The model uncertainty $\sigma$ is on the one hand very high at the beginning of the process, since not much process knowledge by means of observables is available. On the other hand there is not much variance in the state variables at this point, because the impact of different applied blank holder forces is not yet strong, but will play a more important role with increasing time. Although the model uncertainty is very high for time frame number 1, the prediction result is still characterized by a high quality index due to the low variance in the process state. The uncertainty decreases with increasing time, but then also complexity grows and has a stronger impact on the prediction result. The overall model quality $R^2$ as well as the relative error distribution show that the predictions are in good agreement with the original data.

## V. Conclusion and Future Work

It has been shown that the statistical process model for hidden state observation applied to a deep drawing process can be successfully used for process state prediction based on observations. The results outlined in Section IV are very promising and can therefore be taken as a solid base for process control. Process parameters can thereon be adjusted by observing the evolution of the process state implementing a suitable control law. The control of one single process can be extended to process chain optimization by multiple linked process controls. For this purpose workpiece properties are to be deduced from the final state of the final process serving as set value for the optimization procedure. One drawback of the statistical process model is the high uncertainty in state prediction at the beginning of the process. This can be overcome by not considering those early predictions with high uncertainty in process control and by further extensions of the presented approach. Significant time frames for the observation of the process state evolution have to be identified to enable process control. The proposed approach for observation of hidden states for adaptive process controls can be transferred to any process characterized by state variables that can be extracted from related observable quantities.

## Acknowledgment

This work has been supported by the DFG Research Training Group 1483 "Process chains in manufacturing". The authors would like to thank the ITM at the KIT for providing the finite element deep drawing model.

## References

[1] C. Blaich and M. Liewald, "Detection and closed-loop control of local part wall stresses for optimisation of deep drawing processes," in *Proceedings of the International Conference on New Developments in Sheet Metal Forming Technology*, Fellbach, Germany, 2010, pp. 381–414.

[2] Y. Song and X. Li, "Intelligent control technology for the deep drawing of sheet metal," in *Proceedings of the International Conference on Intelligent Computation Technology and Automation*, Los Alamitos, CA, USA, 2009, pp. 797–801.

[3] J. Zhao and F. Wang, "Parameter identification by neural network for intelligent deep drawing of axisymmetric workpieces," *Journal of Materials Processing Technology*, vol. 166, pp. 387–391, 2005.

[4] S. K. Singh and D. R. Kumar, "Application of a neural network to predict thickness strains and finite element simulation of hydro-mechanical deep drawing," *The International Journal of Advanced Manufacturing Technology*, vol. 25, no. 1, pp. 101–107, 2005.

[5] A. Brahme, M. Winning, and D. Raabe, "Prediction of cold rolling texture of steels using an artificial neural network," *Computational Materials Science*, vol. 46, pp. 800–804, 2009.

[6] H. K. D. H. Bhadeshia, "Neural networks and information in materials science," *Statistical Analysis and Data Mining*, vol. 1, no. 5, pp. 296–305, 2009.

[7] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Proceedings of the International Joint Conference on Neural Networks*, Washington D.C., USA, 1989, pp. 593–605.

[8] M. T. Hagan, H. B. Demuth, and M. H. Beale, *Neural network design*. University of Boulder, Colorado, USA: Campus Publication Service, 2002.

[9] W. C. Carpenter and M. E. Hoffman, "Selecting the architecture of a class of back-propagation neural networks used as approximators," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 11, pp. 33–44, 1997.

[10] C. Zhao, F. Wang, N. Lu, and M. Jia, "Stage-based soft-transition multiple PCA modeling and on-line monitoring strategy for batch processes," *Journal of Process Control*, vol. 17, no. 9, pp. 728–741, 2007.

[11] J. Chen and K.-C. Liu, "On-line batch process monitoring using dynamic PCA and dynamic PLS models," *Chemical Engineering Science*, vol. 57, no. 1, pp. 63–75, 2002.

[12] C. M. Bishop, *Pattern recognition and machine learning*, 2nd ed. Springer, 2007.

[13] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed. Springer, 2009.

[14] H. Motulsky and A. Christopoulos, *Fitting Models to Biological Data using Linear and Nonlinear Regression - A practical guide to curve fitting*. GraphPad Software Inc., San Diego CA, 2003.

 ISBN: 978-1-61208-109-0 

# Employing Semantically Driven Adaptation for Amalgamating Software Quality Assurance with Process Management

Gregor Grambow and Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
{gregor.grambow, roy.oberhauser}@htw-aalen.de

Manfred Reichert
Institute for Databases and Information Systems
Ulm University
Ulm, Germany
manfred.reichert@uni-ulm.de

***Abstract*—Often in software development processes, tighter and more systematic integration of quality assurance techniques and measurements in the operational processes is desirable. While some processes specify abstract quality assurance measures, concrete requisite measures directly relevant for specific product artifacts (e.g., code) or processes (e.g., testing) must be determined operationally and contemporaneously, yet are hitherto often determined manually and unsystematically. By employing semantically driven adaptation of software quality assurance and software development processes, an approach is described and applied in the software engineering domain for the detection, mediation, and management of just-in-time quality measure assignments. The approach shows promise for adapting automated process enactment to enable effective and efficient quality assurance integration.**

***Keywords-Adaptive Process Management, Semantic Technology, Software Quality Assurance, Process-Centered Software Engineering Environments***

## I. INTRODUCTION

Software quality assurance (SQA) is often viewed as an area incurring additional costs and delays in delivery of a software project. As cost, functionality, quality, and delivery pressures on product development projects continue, software development processes and quality assurance should be examined for opportunities in the areas of greater automatability and cost reduction. Automation does not only reduce labor costs, but can also systematically improve product and process quality by means of repeatability, traceability, and consistency.

However, automation also raises challenges. Especially the software development domain manifests these due to its dynamic flux in technologies, tools, and platforms that often result in one-off projectized software engineering environments. Process-Centered Software Engineering Environments (PCSEEs) [11] model and execute processes that coordinate human activities and software development tools. Due to the significant impact of SQA on project costs [2], the cost-effectiveness of concrete quality assurance measures (i.e. actions) is essential, and should be based on contextually relevant SEE data for economical adaptation to changing environments and constraints. The timely and harmonious assignment of SQA measures requires their tight integration in software development processes, specifically concrete workflows that must necessarily be adapted since the exact measures are not foreknown but depend on the given context. Due to time constraints and efficiency, only a limited number of all possible quality measures can be applied, while the effectiveness and efficiency of a specific quality measure depends, among other factors, on the applicability of the measure, the project timing, worker competency, and correct fulfillment [2].

One promising area for increasing the degree of automation in PCSEEs is the context-sensitive triggering and selection of quality measures (e.g., refactoring) while taking into account people, processes, tools, artifacts, and interactions. It would include the analysis of quality metrics and measure selection for process and product artifacts to support software engineers via automated measure suggestion at the appropriate time. Moreover, SQA is proportional relative to overall project size [1], while the amount of effort allocated to SQA should be front-loaded (up to 25% of development effort) and be reduced later [2][30]. Manual enforcement of such lifecycle-time-dependent SQA overhead can be laborious and detract from more pressing project management issues.

Software engineers currently lack automated guidance for software quality assurance that can practically apply SQA cost-effectively using SQA models of development activities such as [2]. A solution is required that can analyze the project context on the fly, prioritize SQA measures, and adaptively assign these to the relevant resources at appropriate points in their process. Furthermore, multiuser measures that require preparation (e.g., code inspections) should be contextually coordinated, meaning their activities included at the appropriate point in the user's process. SQA measures should be systematically distributed, and quality measures should be filtered based on the available time constraints. This should be done without exceeding SQA expenditure limits while taking advantage of quality opportunities such as early activity completion where, for example, a suitable measure can be applied without impacting the schedule. This necessitates adaptive processes that can cope with the inclusion of unforeseen activities.

This solution contributes to a holistic PCSEE integration of SQA and SPM (software process management),

 ISBN: 978-1-61208-109-0 

comprising the detection of problems, the automated strategic adaptation of measures in conjunction with the Goal Question Metric technique (GQM) [3], context-based selection of SQA measures, leveraging quality opportunities, and adapting concrete process instances. Measures vary in their effectiveness (i.e. utility) and appropriateness, and measure assignment has to consider these factors.

The remainder of this paper is organized as follows: the solution approach is described in Section II and Section III elucidates the implementation thereof. The evaluation in Section IV is followed by a discussion of related work in Section V, with Section VI providing a conclusion.

## II. SOLUTION APPROACH

The goal of timely assigning quality measures is addressed utilizing a combination of various technologies for detection, mediation, and management, which will now be described. The infrastructure for these technologies is provided by the CoSEEEK framework (Context-aware Software Engineering Environment Event-driven frameworK) [22]. Figure 1 illustrates the conceptual architecture for the current approach.

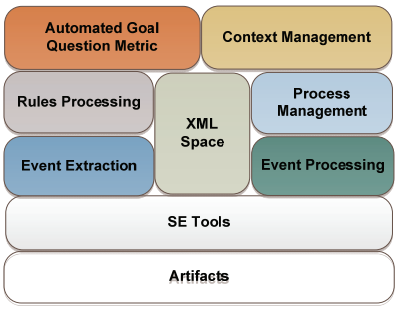



Figure 1. Conceptual Architecture

The architecture of the framework comprises the following components: *Artifacts* is a placeholder for various artifacts that can be processed in a software project, e.g., source code or test code. These artifacts are processed using heterogeneous *SE tools* (e.g., tools for static analysis). Communication and data storage for the event-based loose coupling architecture of the framework are provided by an *XML Space*. To enable integration of diverse external events from SE tools, the *Event Extraction* module is utilized. The *Event Processing* module, in turn, enriches these atomic events with more semantic value through contextual annotation and aggregation of the events based on complex event processing (CEP). A rules engine contained in the *Rules Processing* module automatically analyzes reports from tools for both static and dynamic code analysis, triggering events as necessary. The *Process Management* module applies PAIS (Process-Aware Information System) technology for the automated support and governance of the software development process. Software development processes (e.g., OpenUP) are concretely implemented as workflows in this module. The AGQM (Automated Goal Question Metric) module extends and automates the GQM technique by using a multi agent system (MAS) with behavior agents [10]. Finally, the *Context Management* module enables the aggregation of all information from different sources and thereby the creation of a holistic project context where an adaptive integration of the measures from [10] into the developer's concrete process becomes feasible.

The process of timely assigning apposite quality measures involves several steps as depicted in Figure 2. These are separated into four categories: *Quality Opportunity Detection* detects user availability for quality measures (so called Q-Slots). For this purpose, the estimated and actual durations of planned activities are considered. *Problem Detection / Measure Proposal* comprises activities for automatic detection of problems relating to source code and for the proposal of related quality measures in alignment with higher-level goals utilizing the GQM technique. To facilitate the automated applicability of measures, *Measure Tailoring* comprises three steps for the context-based selection of the measure and its insertion point (so called *extension points* of a workflow) as well as the adaptation of the concrete user's workflow. The procedure is completed by an assessment phase of the applied measures that impacts future measure selection processes, adapting the system automatically while still allowing human-based tuning.

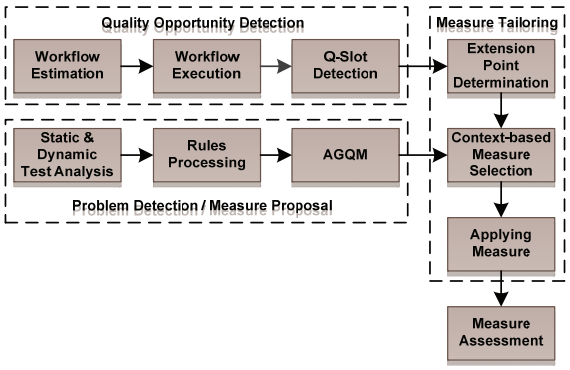



Figure 2. Conceptual Process

### A. Quality opportunity detection

Workflow creation, instantiation, and execution are conducted utilizing the *Context Management* as well as the *Process Management* module. The workflow templates and instances are governed by a PAIS, whereas the *Context Management* module contains semantic enhancements to the workflow concepts. These enhancements include temporal properties, making it possible to estimate durations of activities and match them later with actual execution times. To enable these calculations, an estimation of all durations of user activities (so called assignments) is required prior to workflow instantiation. Other constraints such as planned start date or activity dependencies can also be taken into account. Secondly, the specification of a quality overhead factor and a quality function are required. Via the quality overhead factor, it becomes possible to specify that a certain percentage of work done in a project should be subjected to quality measures. The quality function, in turn, enables control over the timely distribution of that quality effort, e.g.,

 ISBN: 978-1-61208-109-0 

more up-front effort can be allocated to quality measures to reap the benefits described in [2][30]. When a workflow is started, each activity involving a user has a planned duration. Its completion then triggers the recording of the actual execution time. To facilitate the selection of quality measures corresponding to the current situation, the measures are stored in the *Context Module* as well, where they are enhanced with additional properties. These properties comprise an estimation of the duration of a measure and an association of the measure to a certain level of abstraction, meaning a certain workflow such as a project iteration, project phase, or concrete workflow.

Based on these properties, a calculation is applied that reveals *Q-Slots,* i.e. available opportunities for quality measures. During workflow execution, this calculation is conducted each time an activity completes. To determine the actual execution times of the abstract assignments they are connected to concrete tasks a user performs (so-called assignment activities) as depicted in Figure 3. In that example, the assignment activities 'Ac1-Ac5' relate to assignment 'A2' and the assignment activities 'Ac6-Ac8' to assignment 'A3'. The measure proposal process can be started in two possible situations:

1. If activities finish earlier than expected, it is possible to insert a quality measure without delaying forthcoming activities. Therefore, the estimated durations of all processed activities are compared to their actual execution times.
2. If a quality overhead is specified and the quality effort of the current user is below the computed percentage, the insertion of a quality measure activity is also feasible.

### B. Problem detection / measure proposal

Quality measures can be of proactive or reactive nature. To be able to assign reactive measures, an awareness of concrete problems is required. Many problems relating to source code can be detected automatically via static code analysis tools and dynamic test analysis. The *Event Extraction* module enables the CoSEEEK framework to be aware of the execution of such tools, and thus to automatically process reports generated by them. These reports contain information about metrics that were applied to source code. The *Rule Processing* module then processes these reports producing enhanced reports that contain violations to predefined thresholds according to the metrics. Furthermore, the reports include information about the artifacts in which the violations occur and about concrete measures the *Rule Processing* module assigned to them. These reports unify and abstract the reports obtained from all integrated analysis tools, containing reactive measures without order or priority. The *AGQM* module is then responsible for measure selection. Violations of metrics in certain artifacts are also recorded in the *Context Management* module. This enables the aforementioned selection of an *extension point* based on problems relating to artifacts processed by a user's future activities.

The *AGQM* module manages both reactive and proactive (or preventative) measures. It utilizes a multi-agent system (MAS) that selects and prioritizes measures by extending and automating the GQM technique as described in [10]. The latter enables the module to analyze the concrete measures in conjunction with abstract project goals, thus facilitating a strategic measure selection. Reactive measures are prioritized in a cooperative bidding process among the agents, whereas proactive measures are proposed using a competitive bidding process. The module has been adapted to the current scenario to output an ordered list comprising reactive as well as proactive measures.

### C. Measure tailoring

To achieve a high degree of utility for quality measures proposed by the system, their applicability to the current situation is essential. The CoSEEEK *Context Management* Module enables the accumulation and utilization of contextual information from different sources (e.g., the type of the measure or the skill level of the user) to ensure measure applicability. Via semantic enhancements, CoSEEEK enables the process engineer to specify certain *extension points* in the workflow where the application of quality measures is feasible. Examples include the end of an iteration or a phase in a project. Figure 3 illustrates this: It contains workflows on three different levels. *Extension point* 'E1' is attributed to a node of the 'Phase' workflow that, in turn, contains an 'Iteration' workflow. A quality activity can thus be inserted after the completion of that 'Iteration' workflow. Various properties can be included that act as a filter for the types of measures applicable at a certain point.

The *extension point* determination process for a concrete workflow instance is started when an activity is completed and either enough spare time for applying a quality measure exists or the quality effort has not exceeded the specified quality overhead yet. The system searches future assignments of the current user for *extension points* and makes a selection based on different properties. For instance, if the user works on a source code artifact for which the system has detected a problem concerning code quality, an *extension point* at that activity can be selected. Thus, two changes to that artifact can be applied whereas the process of checking out, testing, and checking in the artifact is performed only once. This scenario is illustrated in Figure 3. When the user finishes assignment 'A2' and there is the opportunity for a quality activity, the system checks future assignments, i.e., 'A3' in the given case. That assignment has an attributed sub-workflow, which contains the extension point 'E3', which in turn, is connected to the concrete assignment activity 'Ac7'. That activity involves processing of an artifact for which a problem was detected.

To enable better distribution of quality measures over time and avoid the consideration of quality aspects only at the end of the project, each *extension point* is weighted according to its temporal proximity to current time (*imminence*). If no other constraints apply, the next upcoming extension point is selected.

Since the main purpose of the *AGQM* module is to select, propose, and weight measures in alignment to the goals of the project, an additional selection process is necessary to tailor the proposed measure to the current Q-Slot. This is

 ISBN: 978-1-61208-109-0 

done utilizing the semantic properties of measures contained in the *Context Module*. These properties include the abstraction level to which a measure applies and a time category. The abstraction level facilitates the selection of measures matching the workflow abstraction level, which is related to the selected extension point. The time category enables the selection of measures that consume approximately as much time as the current Q-Slot provides.

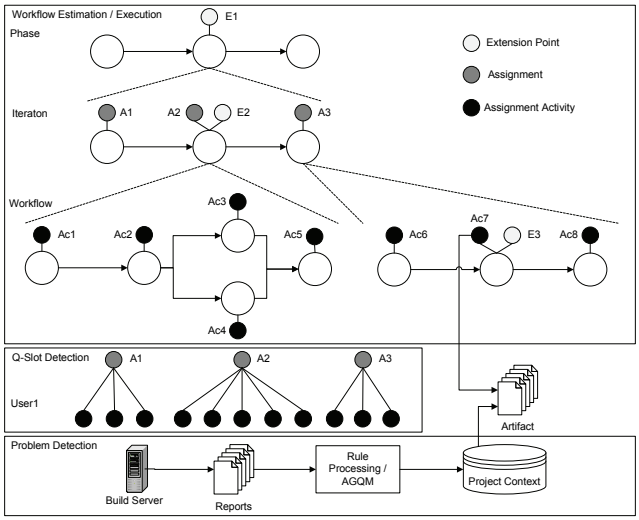



Figure 3. Workflow Enhancements / Q-Slot / Problem Detection

When the Q-Slot is detected, the *extension point* is selected and the appropriate measure is chosen. Further, an activity is generated and dynamically inserted in the workflow of the related user.

### D. Measure Assessment

To further enhance usefulness of the applied measures, an assessment phase takes place after the successful application of the measures. In that phase, the measures are rated due to their impact on the qualities of the produced code. This is done using KPIs (key performance indicators), composite metrics that are continuously calculated by the *AGQM* module. The calculation is conducted each time a report of a testing or analysis tool is received by the system. The KPIs, their values, and the time of their computation are then stored in the *Context Management* module. Based on these values, the measure assessment process is carried out at certain points of time in the project, which can be specified a priori (e.g., at the end of a project or a project phase). The process uses the development of the KPIs over time and the times at which the Q-Slots took place to assess whether or not the measures had an impact on the KPIs. The measures have a usefulness (utility) property, indicating their impact on the KPIs, that is written by the assessment process and used in the measure selection phase.

## III. IMPLEMENTATION

This section describes the technical realization of the different components and concretizes the concept.

### A. Technical realization

The loosely-coupled event-based communication infrastructure for the different modules is provided by a Apache CXF web-service-based implementation of the tuple space paradigm [9] with the eXist XML database [20] for event storage. To facilitate automated problem detection and processing, a combination of the *Event Extraction* module, the *Event Processing* module, and the *Rules Processing* module is utilized. Event extraction is done via the Hackystat framework [13], which provides sensors for various tools. In the current scenario, events from static code analysis tools (e.g., PMD [6]) are used. Atomic events are aggregated using the CEP tool Esper [8]. These events and the reports from the static analysis tools form the input for the *Rules Processing* module that utilizes the rules engine JBoss Drools. Furthermore, that module features a web GUI enabling users to specify code quality measures and relate them to code quality problems. By means of these relations, the module then creates the aforementioned unified reports containing currently violated metrics and attributed measures. Since these are contextually unordered and there can be many violations, the measures are weighted by the *AQGM* module to enable automated measure assignment. That module is implemented using the JADE multi-agent system [4].

Q-Slot detection, context-aware tailoring of measures, and the concrete integration into users' workflows are managed by the *Context Management* and the *Process Management* modules. The Context Management module is realized by an OWL-DL (Web Ontology Language Description Logic) ontology and the semantic reasoner Bossam [12]. The *Process Management* module is based on the AristaFlow BPM Suite [26] for adaptive process support that originated from ADEPT research [7][27]. Due to its 'correctness by construction' principle, AristaFlow supports the efficient and correct modeling, adaptation, and deployment of processes. Its capabilities for ad-hoc workflow changes during runtime, in conjunction with instantaneous checking of correctness, not only enables the safe adaptation of workflow instances, but also offers a high degree of freedom in modeling executable workflows. Workflows allow a process engineer to work with and visualize familiar activity sequences, thus enabling better model understandability, checking, and transfer versus, e.g., a pure ontology (non-workflow) solution. Despite AristaFlow's comprehensive support for dynamic workflow changes, automated workflow adaptation has hitherto not been considered.

### B. Q-Slot detection

Figure 4 shows the concepts in the ontology that enable Q-Slot detection, including one concrete instance (in red) for each class (in black).

The semantic enhancements to workflows are modeled by the concept of the *WorkUnitContainer* containing *WorkUnits* for the activities of a workflow. Since *WorkUnits* and *WorkUnitContainers* respectively do not necessarily have to be related to users, human tasks are modeled separately in the concept of the *AssignmentActivity* that is connected to a *Person*. To allow human estimation of

activity durations prior to workflow execution, the concept of the *Assignment* is introduced. *Assignments* have properties for planned duration as well as for actual execution time. They can have multiple attributed *AssignmentActivities*. An example for an *Assignment* would be 'Develop Feature x' with *AssignmentActivities* like 'Write code' or 'Write Developer Test'. Since the *Assignments* and the *AssignmentActivities* are connected, the actual duration for the *Assignment* can be determined utilizing the durations of the *AssignmentActivities*. Thus, actual spare times for a certain user can be detected by a SPARQL [25] query (which is executed utilizing the Jena API [19]) returning all *Assignments* for that *Person* to compare the planed durations with the actual durations. The percentile quality effort of a user can thus also be determined, comparing the durations of finished *Assignments* with those of processed *Q-Slots*. Using the actual assignment spare time or the available quality overhead limit, if the smallest time category for a measure fits, a *Q-Slot* is generated without an assigned *Measure*. Should another *Assignment* complete before the *Q-Slot* can be inserted, the calculation for the *Q-Slot* is repeated, updating or deleting the *Q-Slot* dependent on available time. Thus, multi-user measures also become possible. If a user generates a *Q-Slot* and another user has already generated a *Q-Slot* but not yet started it, both *Q-Slots* will be connected and multi-user measures be additionally taken into account. The system regards activities such as code inspections as multi-user measures. While these meetings take place out of the scope of the system as part of the users' unestimated activities, any related preparation activities are assigned by the system to the users taking part in a multi-user measure. The scenario section will exemplify the selection of a multi-user measure.

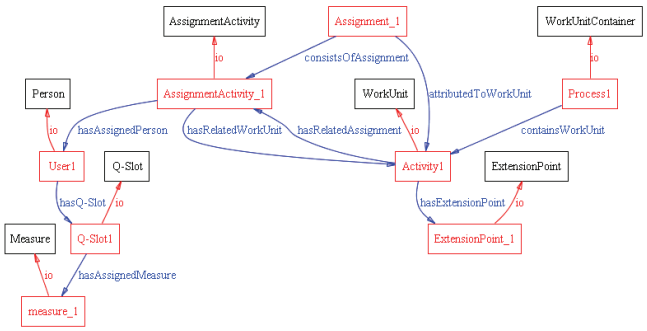



Figure 4. Ontology concepts for Q-Slot detection

## C. Extension point determination

To enable context-based measure selection, possible future *ExtensionPoints* for a user must first be determined. As depicted in Figure 4 *ExtensionPoints* are connected to *WorkUnits* meaning that the *Measure* of a *Q-Slot* can take place after the *WorkUnit* that is connected to the chosen *ExtensionPoint*. To determine upcoming *ExtensionPoints*, the system executes the algorithm depicted in Listing 1, which takes an ordered list of future *Assignments* as input.

Listing 1. Extension Point Determination

```
For assignments
  getRelatedWorkUnit()
  checkForSubExtensionPoints(workUnit)
  checkForSuperExtensionPoints(workUnit)

checkForSubExtensionPoints(workUnit)
  check for extension point and add to list
  if(workUnit has subWorkUnitContainer)
    for each contained workUnit
      checkForSubExtensionPoints(workUnit)

checkForSuperExtensionPoints(workUnit)
  if(finalWorkUnit in WorkUnitContainer)
    getSuperWorkUnit
    check for extension point and add to list
    checkForSuperExtensionPoints(workUnit)
```

Since the *Assignment* can be connected to the *WorkUnit* at any level of abstraction, there can be *WorkUnits* on a level above and below it. The example depicted in Figure 3 illustrates that fact. If there was time for a measure after completion of *Assignment* 'A1', *ExtensionPoints* 'E1-E3' would be available, where 'E1' is one level above and 'E3' is one level below the *Assignments*. For each *Assignment*, the algorithm first checks recursively whether there are *ExtensionPoints* below the current *WorkUnit*. Subsequently, it recursively checks whether the current *WorkUnit* is the final one in its *WorkUnitContainer*. For this case, it checks the superordinate *WorkUnit* to which the *WorkUnitContainer* belongs for an *ExtensionPoint*. The output of the algorithm is a timely ordered list of *ExtensionPoints*.

## D. Context-based measure selection

The context-based measure selection and the final *ExtensionPoint* selection both depend on a weighting of the measures. Prior to that, however, for each *ExtensionPoint,* a query is executed returning the *Measure* with the highest position in the list of the *AQGM* module that matches the properties of the *ExtensionPoint* and the *Q-Slot*. Listing 2 shows the SPARQL query using all of these properties.

Listing 2. Measure preselection

```
PREFIX project:
<http://www.htw-aalen.de/coseeek/context.owl#>

SELECT ?measure
WHERE
 {?list project:containsMeasure  ?measure;
        project:currentList   "1".
  ?measure project:timeCategory  "2" ;
           project:forNumberOfUsers  "1" ;
           project:hasMeasureType ?measureType .
  ?meaureType project:title "MeasureType_1"
  ?measure project:forAbstractionLevel
           ?abstractionLevel.
  ?abstractionLevel project:title
                   "AbstractionLevel_1".
  }
LIMIT 1
```

The properties are the abstraction level, applicable measure type, and user skill level for the *ExtensionPoint* (which do not all have to be set except the abstraction level) together with the time category and number of users for the *Q-Slot*.

The weighting depends on different factors that are all considered: first, the imminence of the *ExtensionPoint* (Imminence *i*: $0 < i <= 1$; initial value 0.9, for each following *ExtensionPoint* 90% of predecessor) is considered. Second is the strategic alignment *sa* of the *Measure* (i.e., the position in the ordered list of the *AGQM* module). Third is the utility of the *Measure* (Measure Utility *mu*: $0.5 < mu < 1.5$; initialized with 1) and defined in Formula (2), which is always updated in the measure assessment process. The fourth factor depends on the measure type and permits weighting of certain measure types (Measure Type Factor *mtf*: $0.5 < mtf < 1.5$; predefined, standard value 1).

An example for this is the measure type 'code', which denotes measures related to source code problems, and thus only can be applied if future activities of the current user involve artifacts for which problems have been detected (such as the activity 'Ac7' in Figure 3). These measures improve efficiency since they can be applied right after scheduled changes to the associated artifacts, avoiding additional overhead for checking out / in and testing. After this weighting procedure, the *ExtensionPoint* with the *Measure* having the highest weight is chosen for insertion. The calculation of the Measure Weight *mw* is shown in Formula (1). For the calculation of the imminence, the index for each upcoming *ExtensionPoints* is n, stating with 0 for the next *ExtensionPoint*. The strategic alignment is computed via the number of measures in the *AGQM* list (listItems) and the position of each measure in the list (listPosition).

$$i = 0.9^{n}$$
$$sa = \left(1 - \frac{listPosition}{listItems}\right) \quad (1)$$
$$mw = i * sa * mu * mtf$$

## E. Applying measures

After all parameters have been determined, the point in the workflow where the activity insertion process shall be started is stored in the ontology as parameter of the *WorkUnit*. The activity is not inserted immediately because if the insertion point lies some time ahead, it is possible that not enough time is left for the measure due to delayed activities processed in the meantime. When the point is reached and there is sufficient time, two possibilities for measure integration exist: initiation of a new workflow instance or direct integration of one or multiple activities in the current workflow. Figure 5 illustrates the insertion of an activity into a running workflow instance using AristaFlow where the green node represents the current activity of the user.

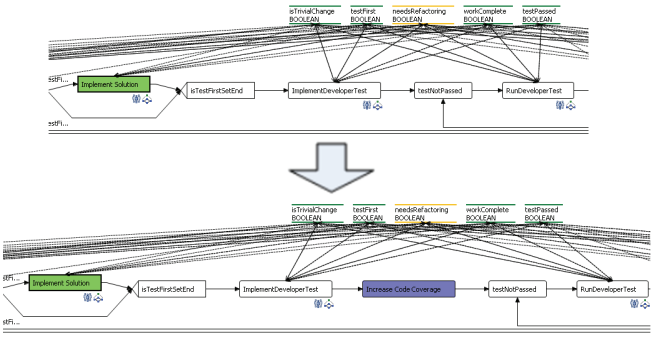



Figure 5. Activity Insertion

The automatic insertion process utilizes the adaptability features of AristaFlow. This allows for seamless integration of the activity into the workflow, while not distracting the current user from his work. AristaFlow also guarantees structural correctness before and after a dynamic insertion.

## F. Measure Assessment

After a measure is applied, related information is stored in the ontology via the *Q-Slot*, which records the time and duration as well as the applied measure and the assigned person. The timepoint of the measure execution is then associated to the impact on KPI trends for utility analysis. Figure 6 depicts a subset of the concepts in the ontology and some concrete instances illustrating the connections between the concepts.

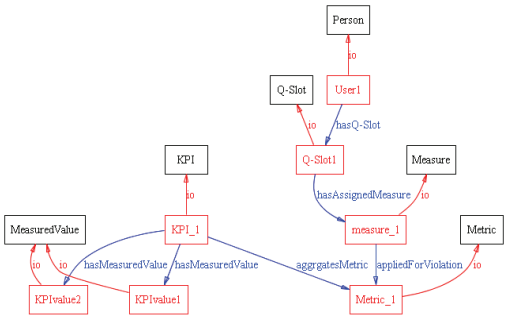



Figure 6. Measure Utility Assessment

*KPIs* are calculated continuously by the *AGQM* module. Each *KPI* has multiple *MeasuredValues* that record the value of one *KPI* calculation at a certain point in time. The assessment procedure works as follows: from the point of time when a measure is applied, up to 10 *MeasuredValues* of the *KPIs* are selected for trend analysis using *KPI* deltas. Since a multitude of factors can influence evolution of the *KPIs,* and since the *Measures* can be of a diverse nature, all 10 values of each *KPI* are used with decreasing influence. The skill level of the person applying the measure is also taken into account, reflecting the higher probability of ineffectively applying a measure by less skilled persons (Skill Factor *sf*: $0 < sf < 1$; predefined, 1 for highest skill level). The equation for calculating *mu* is shown in Formula (2) where n is the index for the *KPI* deltas starting with 0 for the first delta.

 ISBN: 978-1-61208-109-0 

$$mu = mu * \left( 1 + \frac{\sum_0^9 kpi\Delta_n * \left(1 - \frac{n}{10}\right)}{10} \right) * sf \qquad (2)$$

### G. Modeling Effort

To keep the modeling effort reasonable, some functions and standardized definitions are provided. The semantic enhancements to process management (*WorkUnitContainers*, *WorkUnits*, *AssignmentActivities*) are generated automatically from the workflow templates of the *Process Management* module (i.e., AristaFlow). Thus, only the level of the *Assignments* has to be explicitly defined or can be imported from an external work breakdown structure. The connections between the *AssignmentActivities* and the *Assignments* are then automatically established by the system.

A set of standard quality measures can be provided including parameters for common static analysis tools such as PMD or FindBugs. Therefore, the quality manager only has to define the *ExtensionPoints* for the processes and can adapt the values for certain *Measures* if needed. The data provided also includes a standard set of *KPIs* used for *Measure* assessment and for the *AGQM* module, which can also be adjusted.

## IV. EVALUATION

This section covers scalability and performance measurements and a concrete laboratory scenario exemplifying the application of the proposed approach. In planned future work, the validity of the approach will be additionally evaluated in multiple longer-term software production case studies.

### A. Scenario

To illustrate the application of the concept, the OpenUP [23] software development process was chosen with the scenario focusing on the Construction Phase. The 'Develop Solution Increment' process was planned. Figure 7 illustrates the configuration for this scenario.

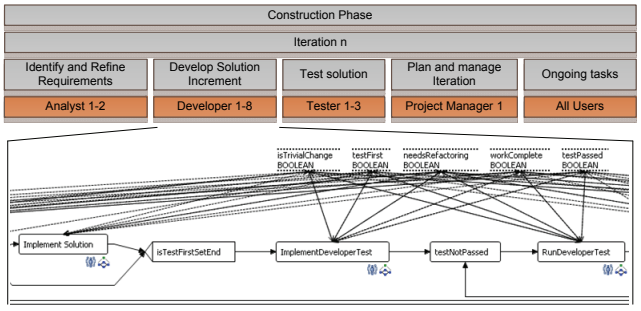



Figure 7. Scenario Configuration

There are four different scheduled and estimated activities *(Assignments*) that are performed by different project members plus activity 'Ongoing Tasks' that represents the base load of each user (i.e., activities which are not part of the project schedule). The scheduled activities are all based on workflows. A snippet of the workflow for the 'Develop Solution Increment' activity is shown in Figure 7 that contains multiple *AssignmentActivities* for the developer such as 'Implement Solution', 'Implement Developer Test', or 'Integrate and Build'.

For the construction iteration, four *ExtensionPoints* have been defined. One takes place after the 'Implement Solution' *Assignment,* having the most concrete abstraction level '0' and the measure type 'code' (referred to as 'Code' in the following). The second takes place after the 'Implement Developer Test' *Assignment* with measure type 'test' ('Test'). Both *ExtensionPoints* are directly related to the processed source or test code the user currently works on and are thus only taken into account if measures relating to the source or test code processed by that activities have been proposed by the *AGQM* module. The third *ExtensionPoint* is attributed to the 'Develop Solution Increment' activity enabling quality measures between the estimated activities ('Activity'). The last *ExtensionPoint is* assigned to the iteration and used for quality measures that only fit at the end of an iteration ('Iteration').

In the current scenario, eight simulated developers are involved, each of them having eight estimated 'Develop Solution Increment' assignments. For simplicity, the scenario only relies on execution time deviation and no specified quality overhead to enable *Q-Slots*. Table I shows the actual execution time deviations from what was estimated, where positive values indicate that an activity took longer than estimated, negative values indicate shorter actual execution times, and grey boxes indicate *Assignments* where after the measure proposal process is started for the respective developer. Since the *Q-Slot* detection only relies on the execution time deviations, it is independent of the initial duration of the *Assignments* or the work hours per day. To keep the current scenario simple, it is assumed that all *Assignments* take one workday. As the table shows, some *Assignments* take longer, while others finish earlier. For this scenario, the quality measures that consume the least time take two hours, which makes quality measures possible for developers 2, 3, 5, and 7.

Table I. Assignment Duration Deviations to Plan (hours)

| **Developer** | **Assignment** | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** |
| dev1 | 0 | 1 | -1 | -1 | 0 | 1 | 0 | 1 |
| dev2 | 1 | 0 | 2 | 0 | 0 | -1 | 1 | 1 |
| dev3 | 1 | -1 | 1 | 0 | 2 | 0 | 0 | 0 |
| dev4 | 0 | 1 | 0 | 0 | -2 | 2 | 0 | -1 |
| dev5 | 1 | 0 | -1 | 0 | 1 | 2 | 0 | 0 |
| dev6 | -4 | 1 | 1 | 1 | 0 | 0 | 0 | -1 |
| dev7 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 0 |
| dev8 | 1 | 0 | 0 | 0 | -1 | -2 | 1 | 1 |

Table II illustrates future *ExtensionPoints* and their values for each user at the point of time where the measure proposal is started for the first time whereas the shaded cells show which *ExtensionPoint* was selected due to the highest

weight. For user 'dev2', this occurs at the end of Activity 3 where measure 'm1' is selected. The value of 0.7 is computed from the following values: $i = 1$, $mu = 1$, $mtf = 1$ and $sa = 0.7$ (the measure being at position 30 in a list of 100 proposed measures from the *AGQM* module). All other 'Activity' *ExtensionPoints* have the same values except for Imminence, which decreases for each additional activity. For the 'Iteration' *ExtensionPoint,* the same values apply except that it has a Strategic Alignment of 0.99 and a Type Factor of 1.2. For user 'dev2', it is assumed that it was detected that (s)he will process a source code artifact for which a problem has been detected as part of Activity 4. Therefore, measure 'm2' is chosen for the 'Code' *ExtensionPoint* of Activity 4. The measure has a Strategic Alignment of 0.75 and a higher Type Factor of 1.4. It therefore is selected for the user.

For user 'dev3', the calculation starts after Activity 5. Since measures 'm1' and 'm3' have not been proposed yet, they are still proposed here due to the same parameters. Thus, 'm3' at the end of the iteration has the higher weighting and is planned for that user.

Table II. ExtensionPoint properties

| Dev | Prop. | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| dev2 | type | A | A | C | A | A | A | A | I |
| | a | 3 | 4 | 4 | 5 | 6 | 7 | 8 | |
| | m | m1 | m1 | m2 | m1 | m1 | m1 | m1 | m3 |
| | w | 0.7 | 0.63 | 0.99 | 0.52 | 0.46 | 0.41 | 0.37 | 0.48 |
| dev3 | type | A | A | A | A | I | | | |
| | a | 5 | 6 | 7 | 8 | | | | |
| | m | m1 | m1 | m1 | m1 | m3 | | | |
| | w | 0.7 | 0.63 | 0.56 | 0.52 | 0.71 | | | |
| dev5 | type | A | A | A | I | | | | |
| | a | 6 | 7 | 8 | | | | | |
| | m | m4 | m4 | m4 | m3 | | | | |
| | w | 0.85 | 0.77 | 0.69 | 0.83 | | | | |
| dev7 | type | A | A | I | | | | | |
| | a | 7 | 8 | | | | | | |
| | m | m1 | m1 | m3 | | | | | |
| | w | 0.7 | 0.63 | 0.79 | | | | | |

A=Activity, C=Code, I=Iteration, a=assignment, m=measure, w=weight, m1=analyze resource usage, m2=refactor code, m3= verify documentation, m4=code inspection preparation

When the calculation for user 'dev5' starts at the end of Activity 6, there is also the Q-Slot of user 'dev3' that has been planned but not used yet. Therefore, two connected *Q-Slots* are available, causing measure 'm4' to be proposed, which is assumed to be a multi-user measure and has a higher Strategic Alignment of 0.85. Because of the higher weight, measure 'm4' is then integrated for users 'dev3' and 'dev5'. Therefore, when the calculation starts for user 'dev7' at the end of Activity 7, measure 'm3' for the 'Iteration' *ExtensionPoint* is still available and is used for that user having the highest weight now.

At the end of the iteration, after all measures have been applied successfully, the assessment process is carried out by the system. This process relies on the development of the KPIs that, in turn, rely on received reports from analysis tools. These reports are generated as part of a nightly build process that builds the code, executes all tests, and applies all metrics to the code in the current scenario. Thus, there are ten points in this scenario where the KPIs are calculated, each taking place after one of the estimated assignments has completed. Table III illustrates the development of the related KPIs. KPIs are computed by the AGQM module to have a uniform value that lies between 0 (extremely bad) and 1 (perfect).

Table III. KPI development

| Measure | KPI | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| m2 | kpi2 | 0.5 | 0.46 | 0.45 | 0.6 | 0.6 | 0.6 | 0.62 | 0.62 |
| m4 | kpi4 | 0.6 | 0.62 | 0.6 | 0.58 | 0.59 | 0.6 | 0.67 | 0.74 |
| m3 | kpi3 | 0.4 | 0.43 | 0.42 | 0.45 | 0.46 | 0.46 | 0.46 | 0.6 |

Since this scenario only covers one iteration with no activities ahead of this iteration, not all ten time points were available for the measure assessment procedure. *Measure* 'm2' was applied at the end of time point 4 but before KPI calculation. Therefore, the first reports of the analysis tools that reflect the impact of that measure are generated at time point 4. Thus, the first delta of the related KPI that is of interest is from time point 3 to time point 4, which is a positive change of 0.15. For Measure 'm4', the first delta used is from time point 5 to 6 and for Measure 'm3' from time point 7 to 8. Since all measures were initialized with a value of 1 for Measure Utility, applying the calculation depicted in Formula (2), the new values for the measures are as follows: 1.1662 for Measure 'm2', 1.154 for 'm4', and 1.14 for 'm3'. The calculations show meaningful values for the synthetic scenario, which does not necessarily mean that the same applies for real world scenarios. Therefore real case studies that will be conducted in the future will not only be used to evaluate the results but also to further refine the calculations.

## B. *Measurements*

Initial performance evaluations were used to assess preliminary sufficiency of the approach for practical use in the field. The critical areas selected are the context-based measure selection and extension point determination in the *Context Management* module and the concrete insertion of activities into a running workflow using AristaFlow in the *Process Management* module. For performance testing, the test system consisted of an AMD dual core Opteron 2.4 GHz processor, 3.2GB RAM, Windows XP Pro SP3, and Java Runtime Environment 1.5.0_20. AristaFlow was used in version 1.0.0beta - r73, its server was running externally in a virtual machine (VM) infrastructure of the university (cluster with VMware ESX server 4.0, 1 GB RAM was allocated for the VM, dynamic CPU power allocation). All measurements were executed five times consecutively using the average of the last three measurements.

The extension point determination and the context based measure selection are conducted together as one part of the whole process. They provide the functionality of the context module. As depicted in Table IV, the measurements were conducted with different numbers of involved *ExtensionPoints*, *Assignments,* and *Measures*. For these measurements, a context was created such that for each measurement run, the number of all involved concepts remained constant.

 ISBN: 978-1-61208-109-0 

Table IV. Context module latencies

| Total number of involved Assignments, ExtensionPoints, Measures | ExtensionPoint Determination (sec) | Measure Selection (sec) |
|---|---|---|
| 5 | 4 | 13 |
| 25 | 19 | 63 |
| 50 | 38 | 125 |

Since the activities inserted in a concrete workflow can consist of multiple nodes, the latency for inserting different numbers of nodes was measured as shown in Table V.

Table V. Process module latency

| Number of inserted activities | Time (ms) |
|---|---|
| 5 | 1052 |
| 25 | 1453 |
| 50 | 2203 |

The measurement results show acceptable scalability for the CoSEEEK approach, as can be seen from the approximately linear increase in computation times.

## V. Related Work

In the area of automation and support for process adaptability, one approach that considers support of users applying ad hoc changes is ProCycle [32]. ProCycle applies case-base reasoning techniques to assist end users in the re-use of process instance changes that were applied in a similar problem context in the past. This way, the quality of process changes can be increased and process learning be fostered. Caramba [33] features support for ad hoc workflows utilizing connections between different artifacts, resources, and processes to provide coordination of virtual teams. The approach presented in [21] utilizes a combination of agent technology and process management technology to enable automatic process adaptations. These adaptations are used to cope with exceptions in the process at runtime. These approaches do not utilize semantic web technology and do not incorporate a holistic project-context unifying knowledge from various project areas as CoSEEEK does.

In related work, several approaches combining semantic and process management technology exist. In [18] a semantic business process repository is presented to automate the business process lifecycle. It features checking in and out as well as locking capabilities and options for simple querying and complex reasoning. An ontology for business process analysis was developed in [24] for improving process compliance analysis for standards or laws like Sarbanes-Oxley act. In [31] the combination of semantic and agent technology is proposed to monitor business processes, yielding an effective method for managing and evaluating business processes. The approach described in [17] aims at facilitating process models across various model representations and languages. It features multiple levels of semantic annotations as the meta-model annotation, the model content annotation, and the model profile annotation as well as a process template modeling language. All of these approaches utilize semantic technology to facilitate the integration of process management into projects. In contrast to this, the CoSEEEK approach does not only use semantic technology to integrate different project areas, but also fosters the automatic and suitable insertion of quality measures into the workflows of project participants.

Various approaches provide integration of the GQM technique into a project. The ISMS (Intelligent Software Measurement System) [5][14] uses different groups of agents for user assistance and determination of different parts of the GQM plan. In [15] a tool was developed that allows the creation of GQM plans using predefined forms as well as the verification of the structural consistency of the plan and the reuse of its components. Furthermore, the tool supports data interpretation and analysis through aggregation of collected data. The approach presented in [28][29] also utilizes agents, for the requirements process of the SW-CMM (Software Capability Maturity Model) model. The focus is the measurement and analysis of software processes using agents and fuzzy logic. cGQM [16] uses the Hackystat framework [13] for GQM, applying continuous measurement with short feedback loops. In contrast to the aforementioned approaches, CoSEEEK focuses on utilizing context knowledge for the automatic adaptation of quality measures based on the GQM technique as well as the adaptation of running workflows for automated concrete assignment of these measures to users.

## VI. Conclusion

Due to its dynamic nature and its adolescence as a discipline, the software engineering domain manifests various challenges for adaptable process and quality automation. Hitherto, development process models have typically remained abstract and not been used for automated executable workflow guidance, while SQA provided abstract quality guidance that relied to a great extent on human triggering, analysis, and concretization of activities. Ideally, opportunities for SQA should be leveraged and applied at the appropriate time with minimal disruption for developers to minimize overhead. Integrating techniques and sensors in the software engineering environment for the operational detection of SQA problems would provide a basis for a context-sensitive mediation of SQA measures and adaptation of automated development process guidance.

CoSEEEK contributes an approach implying semantically-driven adaptation utilizing process management with contextual awareness to diminish the aforementioned challenges. Using sensors and metrics for detecting SQA problems, an extended agent-based GQM technique mediates SQA measures according to KPIs. Opportunities for SQA activities are contextually analyzed, automated measure assignment is adapted, and multi-user measures are coordinated. Adaptable process management enables the system to cope with the dynamicity inherent to SE projects while semantic technology enables contextual adaptation. These result in tighter and timely integration of concretized SQA in the process, improving SQA via the automated prioritization and just-in-time assignment, applying SQA consistently, and reducing SQA overhead by reducing context switches, improving effectiveness by weighing appropriateness (e.g., competencies [2]) and measure utility,

 ISBN: 978-1-61208-109-0 

and detecting and leveraging commonality opportunities (such as artifacts or multi-user). Additionally, the distribution of SQA effort can be adjusted (e.g., frontloading as propagated in [2][30]) and monitored, avoiding issues such as too little too late or overemphasis with the associated negative impact on profitability.

Future work will include case studies in industrial settings, which will be used to assess the applicability and effectiveness of the approach as well as to further refine the model.

## ACKNOWLEDGMENT

The authors wish to acknowledge Stefan Lorenz for his assistance with the implementation and evaluation. This work was sponsored by BMBF (Federal Ministry of Education and Research) of the Federal Republic of Germany under Contract No. 17N4809.

## REFERENCES

[1] Abdel-Hamid , T. and Madnick, S., 'Software Project Dynamics: An Integrated Approach', Prentice Hall, ISBN 0138220409, 1991

[2] Abdel-Hamid, T., 'The economics of software quality assurance: a simulation-based case study,' MIS Q. 12, 3, pp. 395-411, (Sep. 1988)

[3] Basili, V., Caldiera, G., and Rombach, H.D., 'Goal Question Metric Approach,' Encycl. of Software Engineering, John Wiley & Sons, Inc., 1994, pp. 528-532.

[4] Bellifemine, F., Poggi, A., and Rimassa, G., 'JADE - A FIPA-compliant Agent Framework,' Proceedings of the 4th Intl. Conference and Exhibition on The Practical Application of Intelligent Agents and Multi-Agents. London, 1999.

[5] Chen, T., Homayoun Far, B., and Wang, Y., 'Development of an Intelligent Agent-Based GQM Software Measurement System,' doi:10.1.1.146.5373.

[6] Copeland, T., 'PMD Applied,' Centennial Books, ISBN 0-9762214-1-1, November 2005

[7] Dadam, P. and Reichert, M., 'The ADEPT Project: A Decade of Research and Development for Robust and Flexible Process Support – Challenges and Achievements,' Springer, Computer Science - Research and Development, Vol. 23, No. 2, pp. 81-97, 2009

[8] Espertech Event Stream Intelligence. http://www.espertech.com/products/esper.php [June 2010]

[9] Gelernter, D., 'Generative communication in Linda,' ACM Transactions on Programming Languages and Systems, vol. 7, nr. 1, January 1985, pp. 80-112.

[10] Grambow, G. and Oberhauser, R., 'Towards Automated Context-Aware Selection of Software Quality Measures,' Accepted for publication in Proc. of the Fifth Intl. Conf. on Software Engineering Advances (ICSEA 2010). IEEE Computer Society Press, 2010

[11] Gruhn, V., 'Process-Centered Software Engineering Environments, A Brief History and Future Challenges,' In: Annals of Software Engineering, Springer Netherlands, Vol. 14, Nrs. 1-4 / (December 2002), J. C. Baltzer AG, Science Publishers Red Bank, NJ, USA. ISSN:1022-7091, pp. 363-382.

[12] Jang, M. and J.-C. Sohn., 'Bossam: An Extended Rule Engine for OWL Inferencing,' Rules and Rule Markup Languages for the Semantic Web, Springer Berlin / Heidelberg: 128-138, 2004

[13] Johnson, P. M., 'Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System,' Proceedings of the First Intl. Symposium on Empirical Software Engineering and Measurement, IEEE Computer Society, 2007, pp. 81-90.

[14] Junling Huang Far, B.H., 'Intelligent software measurement system (ISMS),' Canadian Conf. on Electrical and Computer Engineering, 2005, pp. 1033 – 1036.

[15] Lavazza, L., 'Providing automated support for the GQM measurement process,' Software, IEEE, Volume 17, Issue 3, May-June 2000, pp.:56 – 62, doi: 10.1109/52.896250.

[16] Lofi C., 'cGQM - Ein zielorientierter Ansatz für kontinuierliche, automatisierte Messzyklen,' Proc. of the 4th National Conf. on Software Measurement and Metrics (DASMA MetriKon 2005), 2005.

[17] Lin, Y., Strasunskas, D., 'Ontology-based Semantic Annotation of Process Templates for Reuse,' in Proceedings of the 10th Intl. Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'05), 2005

[18] Ma, Z., Wetzstein, B., Anicic, D., Heymans, S., and Leymann, F.,. 'Semantic Business Process Repository' In Proceedings of the Workshop on Semantic Business Process and Product Lifecycle Management, 2007

[19] McBride, B., 'Jena: a semantic web toolkit,' Internet Computing, Dec. 2002

[20] Meier, W., 'eXist: An Open Source Native XML Database,' Web, Web-Services, and Database Systems, Springer Berlin / Heidelberg, 2009 pp. 169-183.

[21] Müller, R., Greiner, U., Rahm, E.: 'AGENTWORK: A Workflow-System Supporting Rule-Based Workflow Adaptation,' In Data Knowl. Eng. 51(2), pp. 223-256 (2004)

[22] Oberhauser, R.. 'Leveraging Semantic Web Computing for Context-Aware Software Engineering Environments,' In "Semantic Web", Gang Wu (ed.), In-Tech, Vienna, Austria, 2010.

[23] OpenUp http://epf.eclipse.org/wikis/openup/ [June 2010]

[24] Pedrinaci, C., Domingue, J., and Alves de Medeiros, A.: 'A Core Ontology for Business Process Analysis' (2008), pp. 49-64

[25] Prud'hommeaux, E. and Seaborne, A., 'SPARQL Query Language for RDF,' W3C WD 4 October 2006.

[26] Reichert, M., Dadam, P.,Rinderle-Ma, S., Lanz, A., Pryss, R., Predeschly, M., Kolb, J., Ly, L.T., Jurisch, M., Kreher, U., Goeser, K, 'Enabling Poka-Yoke Workflows with the AristaFlow BPM Suite,' In: CEUR proceedings of the BPM'09 Demonstration Track, Business Process Management Conference 2009 (BPM'09), September 2009, Ulm, Germany, 2009

[27] Reichert, M. and Dadam, P., 'Enabling Adaptive Process-aware Information Sytems with ADEPT2,' In: Handbook of Research on Business Process Modeling. Information Science Reference, Hershey, New York, pp. 173-203, 2009

[28] Seyyedi, M.A., Teshnehlab, M., and Shams, F., 'Measuring software processes performance based on the fuzzy multi agent measurements,' In Proc. Intl Conf. on Information Technology: Coding and Computing (ITCC'05) - Volume II, 2005. IEEE Computer Society, Washington, DC, 410-415.

[29] Seyyedi, M.A., Shams, F., and Teshnehlab, M., 'A New Method For Measuring Software Processes Within Software Capability Maturity Model Based On the Fuzzy Multi- Agent Measurements,' Proc. of World Academy Of Science, Engineering and Technology Vol. 4, 2005, pp. 257-262.

[30] Slaughter, S. A., Harter, D. E., and Krishnan, M. S. 'Evaluating the cost of software quality,' Commun. ACM 41, 8 (Aug. 1998), 67-73.

[31] Thomas, M. Redmond, R. Yoon, V. Singh, R., 'A Semantic Approach to Monitor Business Process Performance,' Communications of the ACM, pp. 55-59, 2005.

[32] Weber, B., Reichert, M., Wild, W., Rinderle-Ma, S., 'Providing Integrated Life Cycle Support in Process-Aware Information Systems,' Int'l Journal of Cooperative Information Systems, 18(1): 115-165, 2009.

[33] Dustdar, S., 'Caramba—A Process-Aware Collaboration System Supporting Ad hoc and Collaborative Processes in Virtual Teams,' in Distributed and Parallel Databases Vol. 15, Issue 1, Kluwer Academic Publishers Hingham, MA, USA (2004)

 ISBN: 978-1-61208-109-0 

# Self-Adaptation in Geotracking Applications: Challenges, Opportunities and Models

Olga Melekhova*, Mohammed-Amine Abchir†‡, Pierre Châtel†, Jacques Malenfant*, Isis Truck† and Anna Pappa†
*Université Pierre et Marie Curie-Paris 6 CNRS, UMR 7606 LIP6
4 place Jussieu, Paris, 75005, France
Email: {Olga.Melekhova,Jacques.Malenfant}@lip6.fr
†LIASD – EA 4383, Université Paris 8
2 rue de la Liberté, Saint-Denis Cedex, 93526, France
Email: {maa,truck,ap}@ai.univ-paris8.fr
‡Deveryware 43 rue Taitbout, 75009 Paris, France

***Abstract*—Geotracking emerges as a mission-critical mean to observe, track and otherwise manage mobile entities from the regular notification of their geographic locations, thanks to positioning devices such as GPS and cell phones. First used in logistic systems, geotracking now strives for large-scale with new applications like the management continent-wide green tax collection on millions of trucks from their passage through thousands virtual toll points. As the current single device, fixed position notification frequency policies can no longer cope with the stringent requirements of such large-scale applications, this paper first examines the challenges and opportunities for self-adaptation in geotracking systems and applications. After identifying the main requirements to that end, it then provides for a first set of decision-making models for identified self-adaptation *scenarii*, in the context of a set of everyday geotracking business-oriented applications. Finally, it proposes a set of tools and the software architecture currently developed under the French ANR project SALTY to address these issues in the conceptual framework of autonomic computing.**

***Keywords*-Adaptive systems; Road vehicle location monitoring; Closed loop systems; Decision-making; Fuzzy control; Global Positioning System; Man-machine interface; Natural language interfaces; Software architecture.**

## I. INTRODUCTION

As geotracking and its use in geographic information systems are more and more prevalent, approaches to optimize their resource usage for given business objectives become a key issue. Typical geotracking applications use periodic position notification from remote positioning devices to track mobile targets. The goal is to follow their progress and enforce geographic constraints such as staying within a predefined corridor or passing through waypoints.

Current practices in geotracking address all of the business objectives using predefined fixed frequencies for devices to push new positions. As each push incurs a cost in data transmission and in device battery usage, fixed frequencies impose quite poor resource usage profiles. Moreover, new programmable positioning devices offer much more flexibility than simply applying a fixed notification frequency. Finally, as geotracking go large-scale, with new applications like the management of continent-wide green tax collection on millions of trucks from their passage through thousands of virtual toll points, striving for adaptive policies and coping with device malfunctions become mandatory.

This paper presents results of the French ANR project SALTY[1] by first exploring the challenges and opportunities in the self-adaptation of geotracking applications. After presenting the conceptual framework underlying geotracking and its self-adaptiveness (§II), the paper takes a bottom-up approach by describing in Section III self-adaptation use cases for a set of exemplary business-oriented geotracking goals: corridor enforcement, waypoint passage notification and delivery point arrival notification. Section IV further develops over the adaptation needs by providing decision models specific to these use cases but general enough to cope with most adaptation requirements in geotracking applications. Requirements and first drafts for a software architecture and accompanying tools are described in Section V.

## II. GEOTRACKING AND SELF-ADAPTATION

Self-adaptation in geotracking strives for a minimal resource usage while sustaining the capability for the system to achieve business objectives, like enforcing a corridor within which a mobile must stay, at some desired level of precision, *e.g.*, notifying the crossing of a corridor frontier with a precision of 500 meters. To better understand how this overall goal can be achieved, this section examines the devices, software platforms and kinds of adaptation that exist in this area.

### A. Positioning devices

Geotracking relies on positioning devices which characteristics deeply impact costs and resource usage. Positioning devices fall into three categories of positioning means:

1) Global Positioning System (GPS) devices are now well-known; they use information and triangulation of signals coming from satellite constellations. Their precision highly depends upon the number of satellites they can "see", the angle between them and even the ground configuration which can cause some bias. After computing

[1]Self-Adaptive very Large disTributed sYstems (ANR-09-SEGI-012).

 ISBN: 978-1-61208-109-0 

positions, devices typically push them to a server using data connection over a GSM network.

2) GSM cell-id uses readily available phone (or similar devices connected to that network) tracking information, *i.e.*, the cell in which the device actually is, to provide estimated positions. Its precision highly depends upon the density of antennas and the geographical form of cell in which the phone is. It needs no specific intervention from the phone itself but is rather a service offered by the cell phone infrastructure. Upon a call to their API, operators provide estimated positions from the geographical locus of the cell containing the phone.
3) WiFi cell-id is similar to GSM cell-id but rather uses WiFi antennas to estimate positions of devices connected to that network. Again, the precision of the method depends upon the density of the antennas.

Devices can be fixed or portable. Indeed, specific GPS devices can be attached to vehicles and be powered by them. Such devices are usually also connected to sensors on the vehicle (door open/closed, motor temperature and speed, fuel level, ...) and can send sensor data along with positions. But more and more portable GPS devices are used for their flexibility, at the expense of running on batteries, which then become a scarce resource. Mobile phones share characteristics with portable GPS, while WiFi cell-id is totally dependent on the device to which the WiFi card is connected.

Most GPS devices send positioning (and possibly other) data through a data link over the GSM network at a frequency that can be dynamically modified by sending them an appropriate command. Many devices can be partly or totally put in sleep mode for economizing energy: the communication subsystem (GSM) can be put in sleep mode between data sendings while the positioning subsystem itself can also be put in sleep mode between signal acquisition and position computations. More and more programmable devices appear everyday on the market. Such devices can embed application specific code that can drive the positioning but also adaptations of this process to some extent.

### B. Geotracking concepts and middleware

One of the key issue in building geotracking applications is how to cope with the diversity of positioning devices. No standard interface exists neither to get positions from devices nor to send them commands. Instead of making end user applications aware of all the possible devices, and have them track down every new devices to be used, a middleware layer can help in abstracting them from this diversity. Deveryware's GeoHub is one such middleware that offers, among other services, a common interface to deal with a large set of different positioning device kinds and models.

Another important issue is the capability to acquire a lot of positioning data and to correlate them in order to capture higher level composite events triggering notifications to end user's applications. Falling under the framework of complex event processing (CEP) [1], indeed geotracking need specialized CEP engines coping specifically with positions and related events. Deveryware's GeoHub central role is to act as such a specialized geotracking CEP, which triggering rules are written in Forth, uploaded to the hub and activated over period of times at customer demand. GeoHub optimizes the process by which positions received from tens of thousands of positioning devices fire end users' notification rules linked to their distant applications.

### C. Potential adaptations

As the basic and prominent functioning of positioning devices consists in sending data at some interval of time, the frequency at which a device must do so can be determined directly from business objectives and the precision required by end users in their notifications.

*Example. Consider the notification of the passage of a vehicle within 500 meters of a waypoint. If the speed of the vehicle is 100 kph at this point, the frequency must be at least one data sent each 36 seconds to make sure a position will be taken within the 1 km diameter around the waypoint.*

But, maintaining such high frequencies while vehicles are far away from any point of interest is just a loss of resources and unnecessary costs. A central adaptation in geotracking is therefore to adjust the frequencies of data sendings so to keep them low when mobiles are far from points of interest but to raise them when approaching such points so to reach the required frequencies to match business objectives with the desired precision. Unfortunately, determining when to raise the frequency requires to predict the position of the vehicle from time to time, to get gradually nearer positions to the objective in order to raise the frequency progressively. Such position predictions depends on a lot of dynamic conditions, like weather, road conditions, traffic, etc. Therefore, dynamic adaptation is mandatory.

Besides this, another important issue is to cope with imprecision or unavailability of devices. GPS positioning precision depends upon the visible satellites and even the ground configuration, which can induce systematic bias, while other means highly depend upon the density of antennas. Switching from one positioning mean to another when conditions require it is another very important adaptation opportunity. As costs of the different mean vary much, GSM cell-id being typically a pay-per-use service from cell phone network operators, keeping their use to a minimum is also an adaptation objective.

Finally, for devices which autonomy on battery is limited, adaptation shall put the device positioning and transmission subsystems in sleep or ready modes back and forth. Putting the transmission subsystem in sleep mode while keeping the positioning subsystem in ready mode taking positions at some frequency can give rise to a form of location-driven positioning instead of the standard time-driven one. Programmable devices can monitor their position and resume sending data only when reaching some target position (or use a fallback rule if the target is missed).

In the SALTY project context, another important adaptation comes from the management of the GeoHub resources. Even

 ISBN: 978-1-61208-109-0 

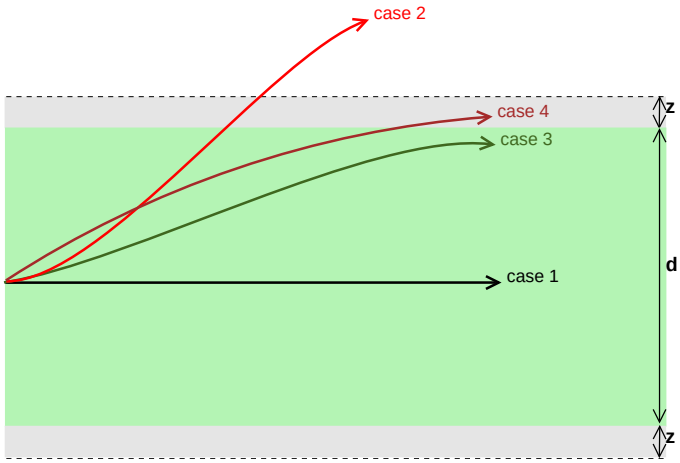



Fig. 1. Different behaviors in corridor enforcement.

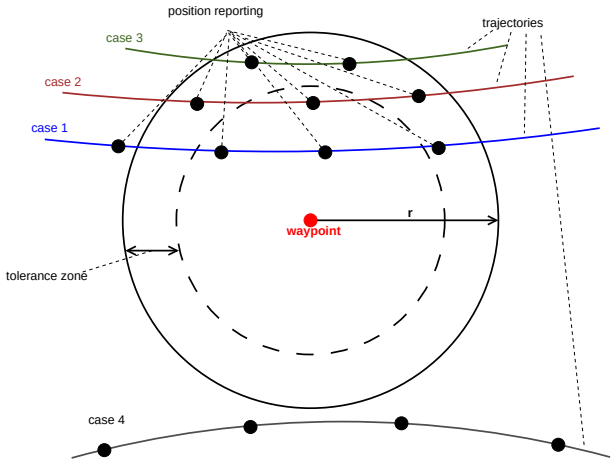



Fig. 2. Waypoint notification tolerance and mobile trajectories.

though more servers can be dynamically allocated to the GeoHub, variations in the workload can still require to adjust more globally the frequencies of mobile data sendings to keep the quality of service in position correlations done by the platform. As the workload of the GeoHub is the sum of the ones generated by every connected device, this kind of adaptation requires to act upon a lot of devices and therefore must scale well when their number grows larger and larger.

## III. Business objectives and adaptation use cases

Geotracking applies to a lot of different areas, such as logistics, people tracking, traffic monitoring, etc. However, Deveryware's experience shows that it revolves around a limited number of similar high level business objectives. This section illustrates them within the context of logistic applications.

### A. Corridor enforcement

A first typical geotracking objective is corridor enforcement, where positions of the vehicle must remain within a short distance from its planned route. End users require to be notified whenever the vehicle goes out of this limit, with a given precision. Typically, the corridor may be d = 10 km wide, and the end user may require a notification of the crossing of a frontier with a precision of z = 500 meters. This can trigger simple route replanning if, after verification, the vehicle happens to try to escape traffic jams, but it can also trigger a more urgent intervention if the vehicle is carrying valuable goods that may just being stolen.

Figure 1 shows in more details *scenarii* in corridor enforcement, where the corridor itself is the green area. The nominal scenario (case 1) is when the vehicle remains more or less in the middle of the corridor, a case where the frequency can be kept relatively low, given the width of the corridor. A second scenario (case 2) is when the vehicle crosses the frontier. The frequency is raised as the vehicle approaches the frontier, and then the crossing of the corridor frontier is notified. The end user may require a tracking of the vehicle after the notification, but typically with less precision, so the frequency may then diminish to adapt to this new requirement

A third scenario (case 3) is when the vehicle approaches a frontier of the corridor, but remains inside it. In this case, a higher frequency will be required to make sure to notify the user with its 500 meters (= z) precision if crossing occurs, and it will be kept as long as the vehicle remains in this situation.

In the last scenario (case 4), the vehicle crosses the frontier but remains within the 500 meters limit outside the corridor. The business rule here says to tolerate these positions, but as it is not nominal, a notification is required if this situation stands still for a too long time (or more relevantly, a combined measure of distance to the frontier and time spent in that situation). Again, the frequency must increase to notify a possible full crossing, or the reaching of the tolerance limit, and maintained as long as this situation persists.

### B. Waypoint passage notification

A second typical geotracking objective is waypoint passage notification. In logistics, waypoints represent meaningful steps when fleet managers want to get feedback to make sure vehicles progress normally on their route, or inquire the drivers if not. In other applications, waypoint notification may serve as basis for green tax collection, as discussed previously.

Figure 2 shows the different *scenarii*. If waypoints are used only to check the progress of the vehicle, the notification may just require to get a position within some radius of the waypoint, *e.g.*, 500 meters, with a tolerance of 100 meters. In this case, it suffices to raise the frequency to a level where we can guarantee that a position will be measured within the circle, at least for a trajectory which passes near the middle of the circle (case 1). Indeed, a higher frequency may be needed for trajectories that will just cut a small part of the circle, as we can see in the case 3, that a fixed frequency will not provide the necessary positions. Overall, the frequency will need to be raised as the vehicle approaches the circle, and maybe to a higher level if the predicted trajectory cuts less of the circle.

 ISBN: 978-1-61208-109-0 

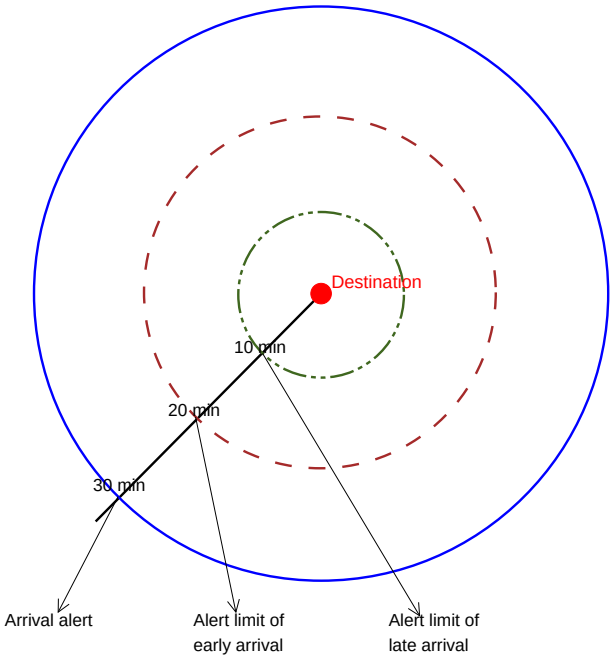



Fig. 3. Delivery point: first notification, early and delayed arrival.

A more stringent scenario comes with applications like the green tax collection. Here, owners of tracked vehicles may require more than one position to be sure the vehicle really passed through the waypoint, to accept to pay the tax. The rule may require three positions within the circle with at most one in the tolerance zone. In such a case, the frequency must be raised much when approaching the circle with a more or less tangent trajectory. In this scenario, being the resource provider for data sendings, the taxation authority may have to balance the cost of raising the frequency against the revenue generated by the tax payment, and decide at some point to give up if the trajectory is as tangent as the case 3.

### C. Delivery point arrival notification

The delivery point arrival notification differs from the waypoint passage by the fact that the vehicle will eventually stop at the destination point. Hence the problem is not to detect a passage but rather to notify the arrival point of the estimated time at which the vehicle must be expected. In logistic applications, such a notification, *e.g.*, 30 minutes in advance, will be used by warehouse dispatchers to choose a door at which the vehicle will be able to stop, deliver (part of) its payload, and take some other payload. As warehouse doors are scarce resources and vehicles and drivers wait time to be minimized (especially when the goods are perishable), a good planning with as much information as possible about incoming vehicles may be a key to success.

Figure 3 illustrates this third business-oriented objective. Typically, a first circle is defined to notify the warehouse of the arrival of the truck. But if the vehicle turns out to progress faster than predicted, an early arrival notification may be required no later than a second limit, *e.g.*, 20 minutes before the predicted time of arrival. Alternatively, if the vehicle is slower than predicted, a late arrival notification may also be required not later than another limit, *e.g.*, 10 minutes before the predicted arrival time.

Adaptation of the frequency follows here a more complex profile. It raises as the vehicle approaches the first limit, in order to notify the arrival with some fixed precision, *e.g.*, 30 minutes more or less 2 minutes. After this first notification, the frequency is reduced, but if the vehicle appears to be too fast or too slow, the frequency is again raised to catch the other time limits with their own predefined tolerance.

### D. Mixed scenarii and other adaptations

Of course, in logistic applications, the three business objectives can be active at the same time. For example, the corridor may still need to be enforced while passing a waypoint or approaching a destination. Adaptation may therefore need to take care of several objectives at the same time and decide for a frequency that matches all of their needs. Other types of adaptations may also be used or may constrain these. For example, putting the device in sleep mode for a while when no objectives are in sight can preserve the battery for a better usage later. Adapting location-driven data sendings discussed before may also be used if programmable devices are available, at least to come to some point nearby the zone of interest. Finally, if problems are detected in the precision of the positions or a device suffers from malfunction, switching to an alternative positioning device may be required.

## IV. Adaptation Model and Decision Making

This section introduces the adaptation model capturing the dynamic of the system in terms of states, decisions, transitions and cost functions to set up de the decision-making problem to be solved to get adaptation policies.

### A. Positioning devices adaptation model

Let $D$ be the set of device types and $P$ be the set of GPS positions, the state of a vehicle can be described by the following state variables:

- $pd \in D$: the type of the primary device used by the truck,
- $sd \in D$: the type of the secondary device used by the truck,
- $ct \in \mathbb{R}$: current time,
- $cp \in P$: the current position of the truck,
- $cf \in \mathbb{R}$; the current frequency of data sendings,
- $bl \in \mathbb{R}$: current battery level,
- $mm \in \{sleep, on\}$: the current measurement unit mode,
- $tm \in \{sleep, on\}$: the current transmission unit mode,
- $sp \in \{ok, failed\}$: the state of the primary device,
- $ss \in \{ok, failed\}$: the state of the secondary device,
- $c$: the corridor imposed to the vehicle,
- $w$: the list of the remaining waypoints,
- $a$: the list of the remaining warehouses to visit.

Each time a decision must be made, this decision will take the form of three components:

 ISBN: 978-1-61208-109-0 

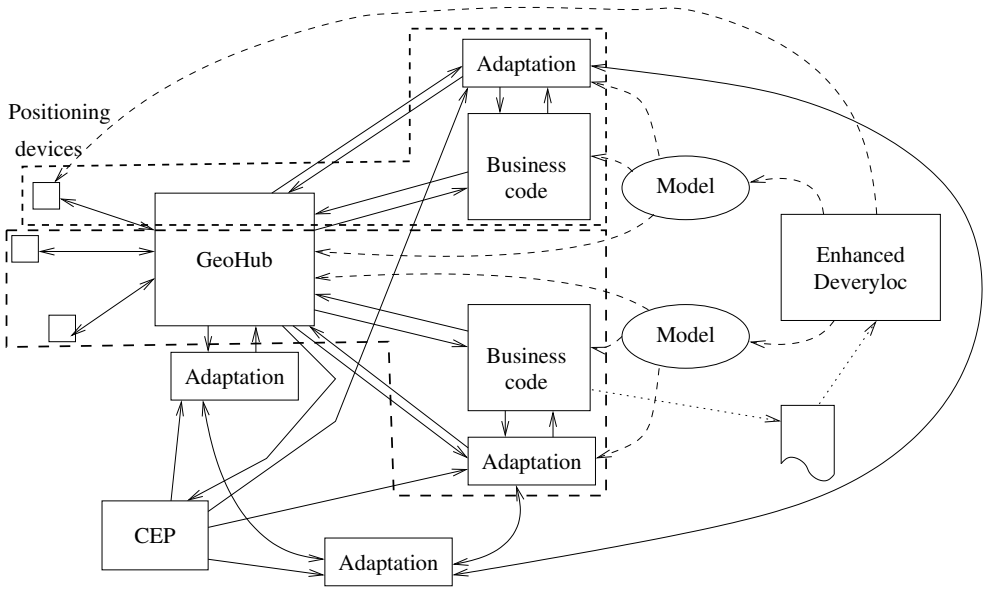



Fig. 4. Elements of the logical architecture.

- $d \in \{primary, secondary\}$: the device that will be used for the data sendings until the next decision,
- $f \in \mathbb{R}$: the frequency at which data will be sent until the next decision,
- $t \in \mathbb{R}$: the time until the next decision, if no failure, battery drained, corridor, waypoints or warehouse arrival event occur.

The cost of transmission is modeled by a function $c_{dt}(t, f)$ giving the total cost of data transmission for a device of type/model $dt$, a time duration $t$ and a steady frequency $f$ over that period. For GSM cell-id, this cost turns out to be the service billed by the operator for calling $n = t \times f$ its positioning API. For GPS devices, this is the cost of transmitting $n$ packets of data over the GSM network.

Energy consumption for battery-powered devices is modeled by a function $e_{dt}(m, m', t, f)$ where

- $dt$ is the type/model of the positioning device,
- $m, m'$ tells if either the measurement or the transmission subsystems are to be put in the new mode $m'$ from a previous mode $m$,
- $t$ is the elapsed time, and
- $f$ is the frequency of measurement and transmission of data (only measurement if the transmission subsystem is in sleep mode).

Transitions to new states are modeled using conditional probability distributions, where $cp$ is the current position:

- $P_c(t|c, cp)$: the probability that the next position in $t$ unit of time will have passed over corridor $c$,
- $P_w(t|w, cp)$: the probability that the next position in $t$ unit of time will have passed the next waypoint in $w$,
- $P_a(t|a, cp)$: the probability that the next position in $t$ unit of time will have passed the next warehouse arrival notification in $a$,
- $P_d(t)$: probability that a failure will occur on a device $d \in \{primary, secondary\}$.

Modeled as above, the goal of the decision-making process is to find an adaptation policy that provides a decision $(d, f, t)$ for each state $s$, which minimizes the cost of data sendings and the probability of a miss of a corridor crossing, waypoint miss or of an unnotified warehouse. Two kinds of algorithms can be applied to compute optimal decision policies: sequential decision making algorithms and greedy algorithms.

When decisions influence each others over time, such as putting a device in sleep mode from which an awakening cost will have to be amortized over the sleep time, algorithms for sequential decision making are required. When the model (cost functions, transition probabilities, and so on) is known in advance, such markovian decision processes can be solved using dynamic programming [2], [3], yielding a policy in the form of a function from states to decisions. Such policies can then be applied at run-time at low cost (in decision making time). When the model elements are not known in advance, run-time machine learning algorithms like Q-Learning [4], [3], are applied.

When decisions do not influence each others over time, greedy, "one-step", algorithms can be applied. The field of optimization offers a lot of such algorithms. As most of the time, the model elements are not known in advance, approaches amenable to run-time learning are preferred. Fuzzy control, introduced by Zadeh [5], proposes such an approach [6].

### *B. GeoHub adaptation model*

As said earlier, if the GeoHub can have its resource augmented to cope with higher workload, these additions of resource come in such large increments that the level of data sendings to it must be controlled to avoid too large decreases in

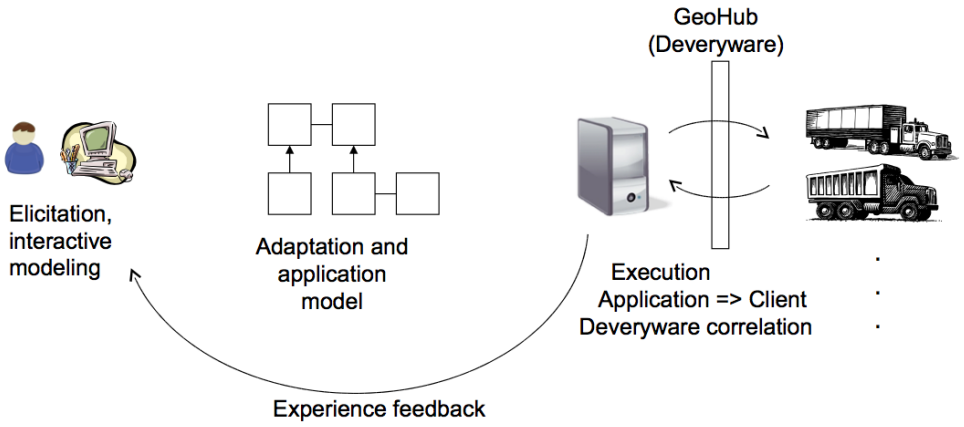



Fig. 5. Overall development steps for geotracking applications.

the quality of service offered by the platform between resource allocations. Controlling the workload of the GeoHub amounts to the control of the frequency of each positioning device currently sending it data. Moreover, as the SALTY project addresses the large-scale, this must be done in a way that scales to a large number of positioning devices. We therefore propose to use decentralized, peer-to-peer algorithms to do so.

When the workload of the GeoHub becomes too large, two kinds of algorithms can regulate it:

- flooding algorithms, which is the diffusion of a command from the adaptation component of the GeoHub to all autonomic managers of devices to reduce their frequency in order to reduce its overall workload, or
- token-based algorithms, where the adaptation component of the GeoHub injects to the autonomic managers of devices tokens representing rights to use some frequency to send data, tokens which will be used and then reinjected towards others when no longer needed.

In the token-based algorithm, a budget of tokens is given to each autonomic manager of positioning devices, which then raises and drops the frequency of its devices within these budget limits. When some autonomic manager requires more tokens because targets approach their next point of interest, thus requiring higher frequencies, a peer-to-peer heuristic algorithm is applied: autonomic managers ask their neighbors, and then these again recursively until one with less current needs than its current budget can give some tokens for a period of time.

## V. Software Architecture and Tools

In the context of the SALTY project, use cases in logistics are developed into an overall logical architecture, shown in Figure 4, and described in the rest of this section.

### A. Application architecture

Applications include business-oriented code with regards to the geotracking objectives and corresponding reactions from a business point of view. For example, this code will be in charge of enforcing a corridor over the route of a vehicle, and if the vehicle exits the corridor, business rules will encode the instructions of the end user to deal with this situation, like calling the driver to know why he did so, and replan a new route if this deviation is justified (traffic jams, accidents, ...).

But applications also include the notification logic and its adaptation. In Figure 4, applications are materialized by bold dashed polylines, which include the business code discussed above, but also the adaptation components. The geotracking notification logic is also part of the application though delegated to the GeoHub and positioning devices. In this vision, the code on programmable devices and the rules loaded into the GeoHub are seen as part of the application, and it is the decision of the programmer as where to put the frontier between what is delegated to these and what will be executed on its own infrastructure. It is one of the objective of the SALTY project to better use programmable devices that appear on the market regularly and promise to lower the costs of geotracking, provided that they are properly used.

Not shown in the figure is the organization of the business code layer, for the sake of understandability. In typical applications, several components will be used to cope with the different business objectives. Typically, components are specialized to specific geotracking objectives, like corridor enforcement. In terms of deployment, each vehicle is usually followed individually, so having one set of specialized components per vehicle is typical of this kind of applications.

### B. Local autonomic management architecture

Each application has its own adaptation layer, which implements the adaptation logic and preferences of the end user. For example, one may put more importance on meeting its objective with a high precision, while another may prefer to keep the overall geotracking costs as low as possible. The basic logical architecture of the adaptation layer follows IBM's autonomic computing blueprint and its Model - Analyse -

 ISBN: 978-1-61208-109-0 

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-model
    uri="http://www.deveryware.com/model/TruckTracking"
    xmlns="http://move.lip6.upmc.fr/geotracking-model/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <alertType name="ArrivalAtWarehouseNotification"
            category="ArrivalAtADestination">
  <application-parameters>
   <param name="Delay" type="xsd:time"/>
   <param name="DelayTolerance"
          type="QualitativeTimeDelayTolerance"/>
  </application-parameters>
  <configuration-parameters>
   <param name="PositioningDevice"
          type="PositionDeviceType"/>
   <param name="StartTime" type="xsd:dateTime"/>
   <param name="EndTime" type="xsd:dateTime"/>
  </configuration-parameters>
  <runtime-parameters>
   <param name="CurrentPosition" type="GPSValue"/>
   <param name="WarehouseLocation" type="GPSValue"/>
  </runtime-parameters>
  <trigger type="FIS">
   <param name="Pos" type="GPSValue"/>
   <fis name="triggerArrivalNotification">
    <fuzzyRule>
     <antecedent type="QualitativeTimeTolerance">
      <two-tuple linguistic-variable="closeTo"
                 displacement="0.0"/>
     </antecedent>
     <consequent type="TriggeringLevel">
      <two-tuple linguistic-variable="High"
                 displacement="0.25"/>
     </consequent>
    </fuzzyRule>
    ...
   </fis>
   <infer fisName="triggerArrivalNotification">
    <expression op="minus">
     <applyFunction name="estimateTravelTime">
      <with-param name="from"
                  value="$CurrentPosition"/>
      <with-param name="to"
                  value="$WarehouseLocation"/>
     </applyFunction>
     <value-of exp="$Delay"/>
    </expression>
   </infer>
  </trigger>
  <emittedEvent
     type="ArrivalAtWarehouseNotificationEvent"/>
  <description lang="en-En">
   When a truck can be predicted to arrive at the
   specified warehouse at the given Delay, the
   specified event is emitted towards the
   application.
  </description>
 </alertType>
 <dataType name="QualitativeTimeTolerance">
  <fuzzyEnumeration>
   <linguistic-variable name="closeTo">
    ...
   </linguistic-variable>
   ...
  </fuzzyEnumeration>
 </dataType>
 <dataType name="TriggeringLevel">
  <fuzzyEnumeration>
   ...
   <linguistic-variable name="High">
    ...
   </linguistic-variable>
   ...
  </fuzzyEnumeration>
 </dataType>
 <dataType
     name="ArrivalAtWarehouseNotificationEvent">
  <data name="TruckId" type="xsd:token"/>
  <data name="occurenceTime" type="xsd:dateTime"/>
  <data name="WarehouseLocation" type="GPSValue"/>
 </dataType>
</geotracking-application>
```

Fig. 6. Models captured through the new DeveryLoc tool in XML format.

Plan - Execute (MAPE) loop [7]. The "monitor" part gets the notifications and positions from the GeoHub, but can also get information about the context from a standard CEP (weather, traffic, etc.). The "analyze" part deals with the decision model presented in Section IV. The "plan" part looks at how to implement the decision, and does the necessary coordination with other related adaptation components. Finally, the "execute" part deals with the actual adaptations acting not only upon the business code component but also on positioning devices and on the GeoHub when required.

To this logical architecture corresponds an implementation architecture, which makes use of several different adaptation components. As several decision making processes apply (adapting frequencies, choosing the positioning mean/device, switching from ready to sleep modes back and forth, etc.), instead of having a unique adaptation component dealing with all of these at the same time, several such components can be implemented, one for each decision problem. These components are then assembled together with an arbitrator composing their independently made decisions into one coherent decision that will be imposed to the base business code component as well as the concerned positioning devices and GeoHub.

Local coordination among tightly-coupled adaptation components is often required as, in general, adaptations have impacts on more than one base element. It is the case here for adaptation components that relate to the specialized geotracking components in charge of one particular vehicle, as they all use the same positioning device. For such coordination among a small number of adaptation components, they can simply be fully interconnected. When the number of components grows, a hierarchical scheme may help in implementing appropriate coordination algorithms.

### C. Global coordination architecture

In some cases, adaptations need to be done at a much larger scale, and even globally. In the architecture of Figure 4, the adaptation component linked to the GeoHub controls its overall workload and performance, as explained in Section IV-B. To support the kind of peer-to-peer coordination algorithms required to go large-scale, the architecture links together the adaptation components managing each positioning device in

 ISBN: 978-1-61208-109-0 

```
<?xml version="1.0" encoding="iso-8859-1"?>
<adaptation-model
   uri="http://www.deveryware.com/model/TTAdaptations"
   xmlns="http://move.lip6.upmc.fr/adaptation-model/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <adaptationType name="TransmissionFrequencyAdaptation">
  <state name="currentPosition" type="GPSValue"/>
  <state name="transmissionFrequency"
         type="xsd:float"/>
  <action name="newTransmissionFrequency"
          type="xsd:float"/>
  <criterion name="getPositionProbability"
             relation-semantics="increasing">
   <function>
    <param name="targetPosition" type="GPSValue"/>
    <param name="targetRadius" type="xsd:decimal"/>
    <expression>
     if (withinRadius(nextPosition(currentPosition,
                                   newTransmissionFrequency),
                      targetPosition, targetRadius))
     then 1
     else if (beforeRadius(predictedPosition(
                                   currentPosition,
                                   newTransmissionFrequency),
                      targetPosition, targetRadius))
     then 1
     else 0
    </expression>
   </function>
  </criterion>
  <criterion name="batteryUsage"
             relation-semantics="decreasing">
   <function>
    <param name="consumptionPerTransmission"
           type="xsd:float"/>
    <expression>
     <value-of exp="$percentagePerTransmission"/>
    </expression>
   </function>
  </criterion>
  <criterion name="totalTransmissionCost"
             relation-semantics="decreasing">
   <function>
    <param name="costPerTransmission"
           type="xsd:float"/>
    <expression>
     value-of exp="$costPerTransmission"/>
    </expression>
   </function>
  </criterion>
  <aggregation>
   <function>
    <param name="getPositionProbability"
           type="xsd:float"/>
    <param name="consumptionPerTransmission"
           type="xsd:float"/>
    <param name="costPerTransmission"
           type="xsd:float"/>
    <expression>
     ...
    </expression>
   </function>
  </aggregation>
 </adaptationType>
</adaptation-model>
```

Fig. 7. Adaptation models captured through the new DeveryLoc tool in XML format.

a self-adaptive mesh striving for an equilibrium between the number of neighbors and the average length of the path connecting any randomly chosen pair of components.

With this kind of mesh in place and correctly maintained by the system, the architecture will efficiently support several kinds of peer-to-peer coordination algorithms:

- flooding algorithms, where the diffusion of a command can be done in time linear in the average path length, or
- token-based algorithms, where the time between the release of a token and its acquisition by another device is also linear in the average path length.

In the context of the SALTY project, the self-adaptive mesh is constructed using heuristic probabilistic algorithms. When a new autonomic component connect its positioning device to the GeoHub, the latter's autonomic component assigns neighbors to the newcomer at random among the already connected ones. Then, these can exchange neighbors with their neighbors if they tend to have more than them. Precise constraints to make this kind of heuristic construction converge to the above good equilibrium are still under study.

### *D. Application and decision-making model elicitation*

The last part of Figure 4, on the right, presents a software tool that completes the logical architecture with an aid for end users to specify their application and eventually connect it to the overall system. Deveryware already offers such a service, called DeveryLoc, which helps end users to configure alerts of predefined types, that is both the triggers and the corresponding notifications, load them onto the GeoHub and start/stop them at will. Currently, this development tool still requires a lot of expertise in geotracking that is rarely available to the clients. Another way to help end users is to propose smart human-machine interfaces to help them specifying their needs. So, the new enhanced DeveryLoc has four major objectives:

1) use the natural language (more precisely a basic discourse analysis) to enhance the HMI in proposing "user-understanding" interfaces;
2) elicit the business-level objectives and their parameters, such as corridor enforcement;
3) elicit the adaptation objectives, such as keeping the frequency low, to save battery or to get a required level of precision in the notifications;
4) enable a qualitative assessment of criteria, instead of a quantitative one, often less intuitive for end users.

The overall needs of the clients are expressed through four different artifacts:

1) an application model defining the kinds of alert types found in the application domain, such as arrivals at warehouse in truck tracking (Fig. 6);
2) an adaptation model defining the kinds of adaptations that can be applied to alerts, such as a position transmission frequency adaptation (Fig. 7);

 ISBN: 978-1-61208-109-0 

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-application uri="http://www.deveryware.com/application/TruckTrackingApp"
                         applicationModel="http://www.deveryware.com/model/TruckTracking"
                         adaptationModel="http://www.deveryware.com/model/TTAdaptations"
                         xmlns="http://move.lip6.upmc.fr/geotracking-prog-model/"
                         xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <alert name="ArrivalAtNiceWN" type="ArrivalAtWarehouseNotification">
    <with-param name="WarehouseLocation"><gpsValue lat="43.7" long="7.26"/></with-param>
    <with-param name="Delay" value="00:30:00"/>
    <param name="DelayTolerance" value="closeTo"/>
  </alert>
  ...
  <adaptation name="frequencyForArrivalAdaptation" type="TransmissionFrequencyAdaptation"/>
  ...
</geotracking-application>
```

Fig. 8. Application description captured through the E-DeveryLoc tool.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<geotracking-configuration uri="http://www.deveryware.com/configuration/TruckTrackingConfig"
                           application="http://www.deveryware.com/application/TruckTracking"
                           type="http://www.deveryware.com/application/TruckTracking"
                           xmlns="http://move.lip6.upmc.fr/geotracking-model/"
                           xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <alertConfiguration alert="ArrivalAtNiceWarehouseNotification">
    <with-param name="PositioningDevice">
      <mobile name="truck1234" xmlns="http://www.deveryware.fr/schema/mobile"
        <phoneNumber>06123456</phoneNumber>
      </mobile>
    </with-param>
    <with-param name="StartTime" type="2010-10-26T08:00:00"/>
    <with-param name="EndTime" type="2010-10-31T18:00:00"/>
  </alertConfiguration>
  ...
</geotracking-configuration>
```

Fig. 9. Application configuration captured through the E-DeveryLoc tool.

3) an application defining alerts from the alert types of an application model to which it applies adaptations from the adaptation model (Fig. 8); and,
4) a configuration defining the exact devices and period of time during which the alerts of an application are activated on the GeoHub (Fig.9).

The separation of concerns exhibited by these different models allows us to define and give different roles in model elicitation. Geotracking specialists can define new application and adaptation models, while end users can define applications and configurations from libraries of existing models.

An example model from the logistics use case, elicited according to the above principles is shown in Figure 6. It corresponds to the models appearing on the right of Figure 4. A model defines alert types, which defines triggering conditions and the notification to be sent. Notice that alert type parameters are divided in three categories:

1) application parameters, that will be provided when defining an application,
2) configuration parameters, given at configuration time, and
3) runtime parameters provided by the GeoHub upon reception of positions to decide whether or not the alert is triggered.

To give a glimpse of the qualitative approach, the trigger of the alert is defined as a fuzzy inference system, based on the process of computing with words [8] here under the 2-tuple [9] representation model. In this case, this approach allows the end user to work with a qualitative tolerance for the time before the truck arrives at the position corresponding to the notification delay in order to trigger this notification. This makes more sense than a precise tolerance, like 2 minutes, given the imprecision of the time estimation made from positions and travel conditions of the truck.

Figure 7 shows the companion adaptation model capturing the kinds of adaptations to be applied to alert types. In this example, the adaptation model defines an adaptation of the frequency of position sendings. Adaptations identify the state information from the system upon which a decision must be made, and the actions that need to be taken. It also defines the criteria, and when there are several ones, an aggregation of these criteria must be fed into the decision process; one possibility is to define an aggregation function computing a single value from the previous criteria.

Figure 8 shows an example of an application. In this example, an alert of the type ArrivalAtWarehouseNotification is defined for the Nice warehouse by simply providing the application parameters to the corresponding alert type. Fig-

 ISBN: 978-1-61208-109-0 

ure 9 then shows the configuration model which provides the configuration parameters for each and every alert in the corresponding application.

With this information, the tool shall generate code skeletons for both the business and the adaptation components, and also for the code to be loaded on the GeoHub and on the (programmable) positioning devices.

As it can be seen in Figure 4, and illustrated by Figure 5, a feedback link from applications to the new DeveryLoc will provide for offline *post-mortem* analysis of application *scenarii* to help end users tailor the parameters of their application needs and resources. For example, if it shows that the application has a too large cost, the end user may adopt less constraining tolerances on the notifications so to lower frequencies, and therefore the geotracking costs. Hence, it is adaptation at another level and under another time frame.

## VI. Related work

The kinds of adaptations, as well as the described software architecture for geotracking, fall into the area of autonomic computing [7], an ambitious goal set by IBM in 2003 of building self-management capabilities in applications. Within this framework, our approach to adaptation falls under the self-configuration and self-optimization concepts, but also under the self-healing, though only using a fallback device.

GeoHub and the set of positioning devices used by the geotracking can be seen as a monitoring system. Hence, the kind of adaptation we have presented can be compared to adaptive monitoring [10], [11], but with dynamic adaptability capabilities [12]. Charbiwala et al. [13] address a problem similar to ours in the context of sensor networks, but they focus on correlation objectives rather than end users' application ones.

Within the field of autonomic computing, feedback control in distributed systems aims at applying control theory and tools to the adaptation of distributed systems, like the rate of sendings in sensor networks [13].

## VII. Conclusion

In this paper, we have explored challenges and opportunities in the self-adaptation of geotracking applications. As geotracking applications go large-scale, a set of resource-aware decision-making models are proposed to strive for an optimal management of positioning devices, system-wide. Moreover, as these applications become more and more mission-critical, fault-tolerance is also being addressed, by integrating the possibility to switch between alternative positioning devices, like fixed GPS blackboxes placed on the vehicle, portable GPS devices running on batteries, and drivers' cell-phones, also running on batteries.

Switching to the geotracking system in the large, the paper also addresses issues in the management of a geotracking hub, used in SALTY to collect and correlate positioning data, as well as notifying end user applications when predefined conditions are met. The use case raises issues in the global management of a large set of positioning devices with regards to their joint use of a shared resource, and paves the way to large-scale coordination for fixing individual positioning frequencies so to minimize the resource usage system-wide, namely the rate of positions sendings coming to the GeoHub.

Having established these use cases and the required decision-making models, future work within the SALTY project include the design of a generic self-adaptive architecture based on the IBM's MAPE loop autonomic computing functional architecture. This generic architecture will be used in the implementation of demonstration applications for the SALTY project. Another important objective is to provide a methodology and a set of tools supporting the development of self-adaptive geotracking applications.

## Acknowledgment

This work is partly funded by the French National Agency for Research (ANR), within the ARPEGE Program, under the project SALTY ANR-09-SEGI-012.

## References

[1] S. Chakravarthy and Q. Jiang, *Stream Data Processing: A Quelity of Service Perspective*. Springer, 2009.

[2] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.

[3] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[4] C. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[5] L. Zadeh, "Outline of a new approach to the anamysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

[6] K. Michels, F. Klawonn, R. Kruse, and A. Nrnberger, Eds., *Fuzzy Control — Fundamentals, Stability and Design of Fuzzy Controllers*, ser. Studies in Fuzziness and Soft Computing. Springer-Verlag, 2006, no. 200.

[7] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[8] L. Zadeh, "The Concept of a Linguistic Variable and Its Applications to Approximate Reasoning," *Information Sciences, Part I, II, III*, vol. 8,8,9, pp. 199–249, 301–357, 43–80, 1975.

[9] F. Herrera and L. Martínez, "A 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 6, pp. 746–752, 2000.

[10] M. A. Munawar and P. A. S. Ward, "Adaptive Monitoring in Enterprise Software Systems," in *SIGMETRICS 2006 Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 2006, http://research.microsoft.com/en-us/um/redmond/events/sysml/papers/sysml-Munawar.pdf.

[11] S. Agarwala, Y. Chen, D. Milojicic, and K. Schwan, "QMON: QoS- and Utility-Aware Monitoring in Enterprise Systems," in *Proceedings of the IEEE International Conference on Autonomic Computing, ICAC'06*. IEEE Computer Society, June 2006, pp. 124–133.

[12] B. Le Duc, P. Châtel, N. Rivierre, J. Malenfant, P. Collet, and I. Truck, "Non-functional data collection for adaptive business processes and decision making," in *Proceedings of the 4th Int. Work. on Middleware for Service Oriented Computing (MWSOC'09)*. ACM, 2009, pp. 7–12.

[13] Z. Charbiwala, S. Zahedi, and Y. Cho, "Toward Quality of Information Aware Rate Control for Sensor Networks," in *Proceedings of the FeBID 2009 Workshop*, 2009, http://controlofsystems.org/febid2009/papers/p11_s24_CharbiwalaZahedi.pdf.

 ISBN: 978-1-61208-109-0 

# Bee Inspired Online Vehicle Routing in Large Traffic Systems

Horst Wedde, Sebastian Senge, Sebastian Lehnhoff, Fabian Knobloch, Tim Lohmann, Robert Niehage, Manuel Sträßer
School of Computer Science
Technical University of Dortmund
Dortmund, Germany
{horst.wedde, sebastian.senge, sebastian.lehnhoff, fabian.knobloch, tim.lohmann, robert.niehage, manuel.straesser}@tu-dortmund.de

***Abstract*—Traffic congestions have been a major problem in metropolitan areas worldwide, causing enormous economical as well as ecological damage. We argue that, due to the highly dynamic character of congestion forming and dissolving, any adequate solution for individual online vehicle routing in large traffic systems will require distributed, adaptive coordination of local navigators in order to transmit directions in due time before any road intersection which would still be valid when carried out. In this paper a completely distributed and adaptive swarm intelligence based multi-layered approach termed BeeJamA is presented. It features dynamic deadlines. There is no need of global or centralized information. We report on extensive simulation experiments with the MATSim simulator verifying BeeJamA's superior performance compared to existing models.**

***Index Terms*—Traffic management; traffic information system; distributed systems; swarm intelligence**

## I. Motivation

Traffic congestions in densely populated areas induce high economical and ecological costs worldwide. The total economic loss in Europe alone, during the year 2000 was estimated to exceed 270 billion euro [1].

For an adequate individual online vehicle routing to be practically successful in large traffic systems it is necessary that driver directions should be transmitted and processed in due time before each intersection, and that they should still be valid when carried out. This calls for dynamic and very tight deadlines. (Otherwise the measures may even be counterproductive.) Also successful solutions should be seamlessly scalable to very large systems, and at the same time, they should be robust considering that when introduced the drivers will only gradually accept the routing system. None of the so far existing approaches satisfies all three constraints. Due to space limitations we have to mostly skip previous and related work, however, it is not difficult to see that any satisfactory model has to be based on distributed agent control and a dynamic management of deadlines in the range of very few seconds. Global information handling would make the system too slow. For coping with the highly dynamic complexity of individual vehicle traffic we developed a novel self-organizing online routing system termed BeeJamA (for *Bee Jam A*voidance). Here autonomous agents (navigators) coordinate their area information through a multi-layer communication structure, the latter relying on our own novel routing protocol BeeHive/BeeAdhoc [9] for large computer networks which had been inspired by the honey bee foraging communication [2], and was stepwise adapted to the road guidance situation. The objectives are congestion avoidance and minimization of individual travel times. After achieving a proof of concept through a fairly simple traffic model [3] we deal with the general case here. A notable related work is the H-ABC [4] , an adaption of AntNet [5] (and thus decentralized approach) to traffic networks, however, for a detailed comparison between BeeHive and AntNet as underlying agent model we refer to [9].

The next sections are structured as follows. Section II briefly outlines the behavior of honey bees in nature. Section III describes our communication and system architecture and Section IV introduces our novel Generic Routing Framework, as the central modeling contribution and at the same time the key for implementing BeeJamA (and other algorithms) into a simulator, and to compare it to traditional algorithms. Section V sketches the main ideas and concepts of the BeeJamA algorithm. In Section VI we report on our simulation experiments with the MATSim [6] simulator and summarize the results in Section VII.

## II. Honey Bees in Nature

A honey bee colony reacts flexibly and adaptively to countless changes in the forage pattern outside the hive, and to changes inside the hive, through a decentralized and sophisticated communication and control system (by watching so-called waggle dances. In this way each honey bee follows simple stochastic rules relying on local information only. A reinforcing and self-regulating behavior emerges in a decentralized fashion where the amount of collected food is optimized in a decentralized fashion, and food sources are not overworked.

The exploitation of such principles by means of a *multi agent system* (MAS) is frequently termed Swarm Intelligence [7]. For a complete description of the biological background, which we used as the foundation of our MAS, see [2].

To sum up, the global optimization problem (of collecting enough food in the natural scenario or of jam reduction in the

 ISBN: 978-1-61208-109-0 

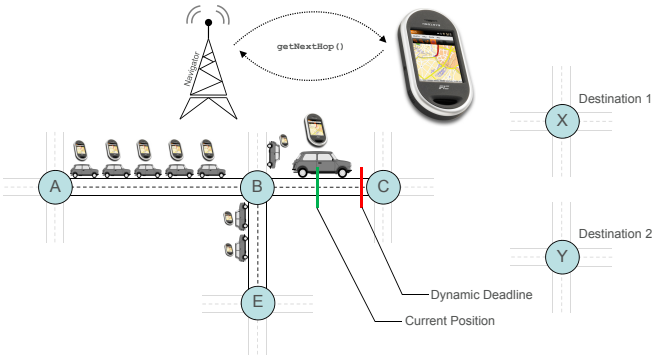



Fig. 1. Dynamic Deadline

road traffic scenario) is tackled by agents solving local and dynamic assignment problems under distributed control. Our algorithm borrows from this decentralized and self-organizing foraging behavior.

## III. V2I Communication

The BeeJamA system uses an approach similar to Floating Car Data (FCD) [8] to get up-to-date traffic information. The BeeJamA algorithm disseminates this information and calculates individual routing instructions based on it.

In this section, we propose a vehicle-to-infrastructure (V2I) based architecture for the necessary communication with the vehicles. The cornerstone of the architecture is a decentralized network of so-called *navigators*. A navigator has a spatially limited area of responsibility, its *navigation area*, where it handles the communication with each vehicle and returns routing instructions on request.

The interaction between a vehicle and a navigator is depicted in Figure 1. After the driver specified a destination, a (GPS-enabled) personal navigation assistant (*PNA*), e.g. a smart cell phone, continuously submits the vehicle's position to its responsible navigator. In addition, a routing instruction request for the next intersection (hop) is sent from the PNA to the navigator each time a vehicle enters a new road. The navigator calculates an instruction (see Section V-D) based on up-to-date traffic information (which, in turn, is based on the continuously transmitted position information of the vehicles and disseminated by a flooding procedure). Ultimately, a hop-to-hop routing emerges, where each vehicle in a navigation area receives an individual next-hop instruction before each intersection in due time. This is a tough real-time problem as these individual deadlines depend on the speed of the vehicle moving within the area of its responsible navigator. Therefore, the size of a navigation area must be small enough to allow for timely detour calculation in the presence of congestions (and it is further limited by the actual wireless communication range).

To cope with these real-time constraints a completely distributed routing algorithm (based on the described decentralized V2I architecture) is necessary to achieve these objectives.

## IV. Generic Routing Framework

In order to conduct simulations easily, we developed a middleware (termed *Generic Routing Framework* (GRF)) to hide the complexity of traffic simulation from the routing algorithms and to offer a common interface to routing algorithms from the simulation tools. In such a way we could easily switch to another simulator, or use a different set of routing algorithms. The GRF manages a graph $G$ (representing the road network) and a set of agents called tokens (representing the vehicles) on the so-called *physical layer*. The physical layer maintains domain specific physical attributes like the length of a road and the position of each token at each time step. For that purpose, the simulator computes the vehicle positions and updates the GRF data structures accordingly (using the appropriate interfaces).

The weighted directed graph is given by $G = (V, E, w)$, where $V$ is the set of vertices, $E$ the set of edges and $w : E \times \mathbb{N} \rightarrow \mathbb{R}^+$ a time variant weight function (the second parameter $t \in \mathbb{N}$ represents a discrete time step). Edges represent roads and vertices represent intersections, thus the graph correlates directly with the actual road network. The weight function on the physical layer simply indicates the number of vehicles at time $t$ on edge $e$, or their current travel times, as given by the road traffic simulation tools. The graph is assumed to be strongly connected.

The implemented routing algorithms may add abstract layers to maintain algorithm specific information for the computation of the routing decision, e.g., the BeeJamA algorithm adds two layers as described in the next section.

## V. The BeeJamA Algorithm

The basic idea of our BeeJamA algorithm presented in this section is for the sake of scalability to follow a three layer approach (Figure 2) to allow real-time hop-to-hop routing in road networks of a realistic size. The first layer is given by the aforementioned physical layer. The BeeJamA algorithm has two more layers each consisting of a graph, a set of routing tables, and rules for the generation of bee agents used to disseminate local weight changes.

On the first additional layer, called *navigation area layer*, or *area layer* for short, the graph layout is nearly congruent the physical layer's graph structure but is partitioned into the areas. Each area belongs to a navigator which is responsible for maintaining the routing tables associated to the area, sending and receiving bee agents for the routing table updates and for communicating with the vehicles within the area. A vehicle which travels over larger distances thus traverses more than one area to reach its destination. On the second additional layer, called *net layer*, the information needed for routing between areas (and 'in the direction of the destination area") is managed. The actual techniques are described in the following sections.

On both layers two kinds of routing tables are used: *Next-Hop* (NH) and *Node-to-Hierarchy* (NTH) tables. Table I and II depict generic instances of these tables.

 ISBN: 978-1-61208-109-0 

TABLE I
STRUCTURE OF A NEXT HOP TABLE

| $NH(v)$ | $W_1$ | $\cdots$ | $W_{\lvert W\rvert}$ |
|---|---|---|---|
| $S(v)_1$ | $c_{1,1}$ | $\cdots$ | $c_{\lvert W\rvert,1}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $S(v)_{\lvert S(v)\rvert}$ | $c_{1,\lvert S(v)\rvert}$ | $\cdots$ | $c_{\lvert W\rvert,\lvert S(v)\rvert}$ |

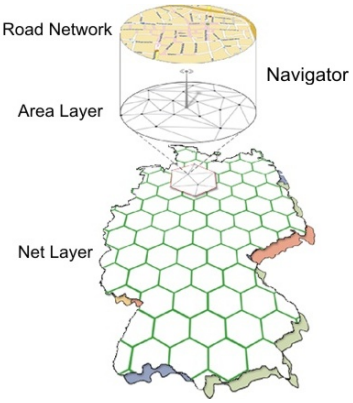



Fig. 2. Layered Routing Model

Let $S(v)$ denote the set of successors of node $v \in V$ and $W \subseteq V$ a set of destinations. Then, the cost to travel from node $v$ to $W_i$ over successor $S(v)_j$ is $c_{i,j}$ in a next hop table. The actual set $W$ differs on both layers.

TABLE II
STRUCTURE OF A NODE-TO-HIERARCHY TABLE

| $NTH$ | $W_1$ | $\cdots$ | $W_{\lvert W\rvert}$ |
|---|---|---|---|
| | $H_1$ | $\ldots$ | $H_{\lvert W\rvert}$ |

The node-to-hierarchy table is a simple associative memory, mapping nodes bijectively from a set $W$ to sets $H_1, \ldots, H_{|W|}$. The elements of $H_i$ are structural components like the aforementioned navigation areas. Such node-to-hierarchy tables are static and do not change throughout the simulation. In contrast, the table entries of next hop tables are time variant and are updated by incoming bee agents (see Section V-C).

### A. Area Layer

The basic notion of our distributed routing algorithm is to divide the physical layer graph (i.e., the actual road network) into smaller parts called *navigation areas* (done on the area layer) and to connect them to allow inter-area travels (done on the net layer). So the navigator associated to each area could refrain from maintaining global information and must thus maintain only small routing tables with local (and therefore up-to-date) information for intra-area routing and information about a limited neighborhood for inter-area routing on the net layer. Technically, the area layer's set of vertices and edges are equal to their counterparts on the physical layer, $G_A = (V_A, E_A) = (V, E)$. The weight function $w_A$, however,

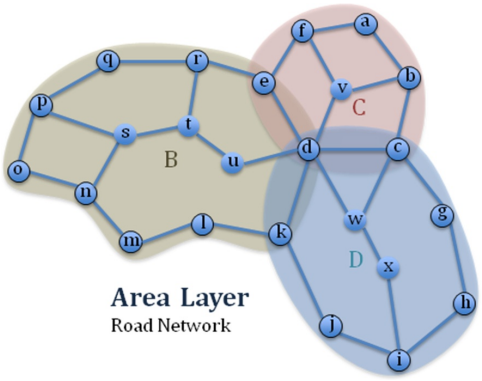



Fig. 3. Area Layer

differs, since it transforms the information available on the physical layer into (estimated and thus abstract) travel times for each edge. E.g., we use a simple moving average of the last $n$ travel times over an edge (given by the simulator) to determine an edge's weight. An area is defined using an edge partition of the area layer's graph. Thus, given an edge partition $E_{A,1}, \ldots, E_{A,n}$, an area is a subgraph $A_i = (V_{A,i}, E_{A,i})$ with

$$V_{A,i} = \bigcup_{(e_k,e_l)\in E_{A,i}} \{e_k, e_l\}.$$

The set of border nodes $B(A)$ of an area $A$ is given by $B(A) = \{b \in V_A | \exists i \in V \setminus V_A : (b,i) \in E\}$ and the set of inner nodes by $I(A) = V_A \setminus B(A)$. Please note, that border nodes belong to at least two areas. At the same time, the set of edges is partioned such that each edge is assigned to exactly one navigation area (e.g., edge $(d,e)$ in Figure 3 belongs either to area $B$ or $C$). We use a simple grid algorithm to obtain an edge partition. Each navigator maintains two tables: for each node $v \in V_{A,i}$ the next hop $IFZ_{area}$ (intra foraging zone) table and a copy of the static and global node-to-hierarchy $FRM_{area}$ (foraging region membership) table. The destination set $W = V_{A,i}$ is utilized for intra-area next hop routing (see section V-D for details on routing decisions). To locate a destination node's area, the navigator checks its $FRM_{area}$ table which maps each node to its associated areas.

### B. Net Layer

Typically vehicles may drive across several areas to reach their individual destinations. To satisfy those routing requests, for each area on the area layer a so-called *net area* on the second layer, the *net layer*, is created. These net areas are mapped onto fixed *foraging regions*, modeling the vicinity of a destination node (see Figure 4 and Figure 5 for examples).

Each foraging region is represented by a *representative node*. In addition, each node $v$ on the net layer maintains a specific *foraging zone* $FZ^r_{net}(v)$ that consists of all neighboring nodes within a certain hop range $r$. It models the direct vicinity of a source node for which accurate routing information is available. They are constructed by a flooding procedure originally developed for the BeeHive algorithm [9].

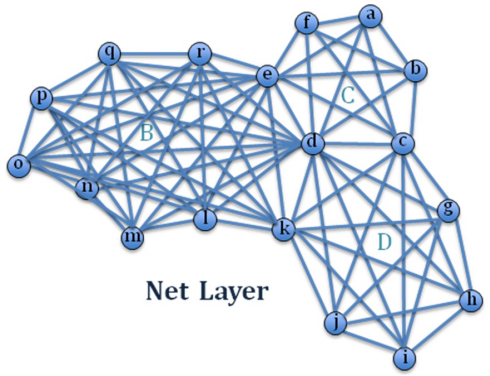



Fig. 4. Extended Net Layer with 3 Areas

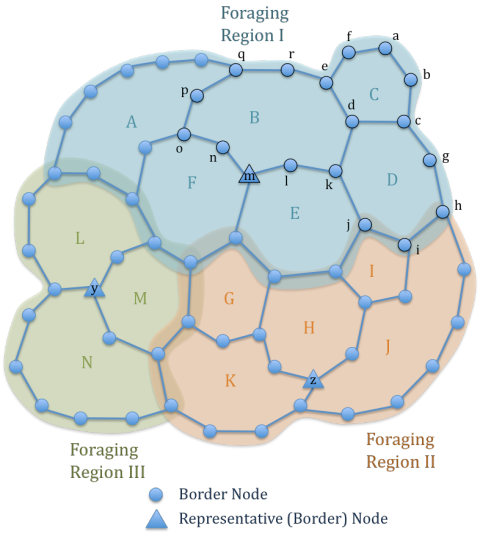



Fig. 5. Net Layer with Foraging Regions

Let $A_1, \ldots, A_n$ be the areas on the area layer, then the net area $N_i$ is given by the fully connected graph $N_i = (B(A_i), E_i)$ with

$$E_N = \bigcup_{i=1}^{n} \{(v_1, v_2) \in E | v_1 \in B(A_i) \wedge v_2 \in B(A_i)\}.$$

The net area's navigator maintains three tables: for each node $v \in B(A_i)$ the next hop tables $IFZ_{net}$ (inter foraging zone) and $IFR_{net}$ (intra foraging region) and a copy of the static and global node-to-hierarchy $FRM_{net}$ table . The $IFR_{net}$ table (in analogy to the $IFZ_{area}$ table ) stores costs from each neighbors to each nodes in the same foraging zone on the net layer ($W = FZ^r_{net}(v)$). In addition, the $IFR_{net}$ table stores costs from each neighbors to each representative nodes and is used to forward vehicles if the destination is far away (see Section V-D for more details) and finally the $FRM_{net}$ table maps each net layer node to its foraging region (and thus to its associated representative node).

### C. Table Updates

In order to continuously update the routing tables, we follow a multi agent system (MAS) approach. We use two types of agents, inspired by the honey bee behavior: the majority of the foragers exploit food sources in the direct vicinity of the hive, while a minority visit food sources which are further away. We adapted this concept into *Short Distance Bee Agents* and *Long Distance Bee Agents*. Both types of agents are responsible for disseminating routing cost information. They only differ in the distance (hops) that they are allowed to travel starting from their launching node. Due to page limitations we refer the reader to a comprehensive description of the MAS based update process in [9].

Here, we present a shortend informal version of this process:

1) Each non-representative node periodically sends a short distance bee agent, by broadcasting replicas of it to each neighbor site.
2) When a replica of a particular bee agent arrives at a site it updates routing information there, and the replica will be flooded again (it will not be sent to the neighbor from where it arrived). This process continues until the life time (number of hops) of the agent has expired, or if a replica of this bee agent had been received already at a site. In the latter case the new replica will be killed there.
3) Representative nodes launch long distance bee agents that would be received by the neighbors and propagated as in 2. However, their life time (number of hops) has a higher limit, the long distance limit.

As a result of this flooding based approach, each node always has up-to-date cost information (depending on the flooding period and the graph layout) of the travel times to the nodes in its own foraging zone and to the representative nodes.

### D. Routing decisions

In the area and net layer model, there are three possible cases for routing a vehicle from node $x$ within area $X$ to a destination node $y$ within a destination area $Y$:

1) The destination node $y$ is listed in the current area $X$'s $IFZ_{area}$ table. Thus, the destination node $y$ lies within the current area $X$ (areas $X$ and $Y$ are the same):

$$\textbf{RC I} : y \in X$$

2) Case 1 is not true but the destination area $Y$ is listed (relative to every border node of $Y$) in the $IFZ_{net}$ tables of the current area $X$'s border nodes. Thus the destination node $y$ lies outside of area $X$ but within an area $Y$ that lies within the net layer foraging zone[1] of area $X$:

$$\textbf{RC II} : \neg\text{RC I} \wedge B(Y) \subseteq FZ^r_{net}(X)$$

3) Case 1 and 2 are not true (i.e., the destination node $y$ lies outside of area $X$ and the destination area $Y$ lies outside of the net layer foraging zone of area $X$):

$$\textbf{RC III} : \neg\text{RC II}$$

[1] $\forall X \subseteq V : FZ^r_{net}(X) = \bigcap_{x \in X} FZ^r(x)$

The cases above reflect a sequence of scenarios in which the destination node $y$ lies further and further away from the current area $X$.

The navigator in $X$ selects the vehicle's next hop by creating a set of *cost paths* $C(x,y)$, evaluating the costs $c(p)$ of each cost path $p \in C(x,y)$ as appropriate in the particular routing case and drawing a cost path $p$ (including a next hop) accordingly. Thus, the decision process in each routing case is described by stating the set of cost paths $C(x,y)$ and the associated cost function $c(p), p \in C(x,y)$. The cost paths[2] $C(x,y) = \left\{\left(x \dot{\rightarrow} S(x) \rightarrow \ldots \rightarrow y\right)\right\}$ are no complete paths from $x$ to $y$ but a sequence of nodes used to approximate the costs of choosing each $s \in S(x)$ as a next hop.

Routing in case 1 is fairly simple. The vehicle's next hop is chosen probabilistically according to the (normalized) costs of the next-hop entries in the $IFZ_{area}$ table of area $X$. The cost path expresses that the destination node $y$ is reachable within the current area by each neighbor $s \in S(x)$:

$$C(x,y) = \left\{\left(x \dot{\rightarrow} S(x) \rightarrow y\right)\right\}$$

Thus, the number of cost paths to evaluate is

$$|C(x,y)| = |S(x)|$$

Each cost path $p = (x \dot{\rightarrow} s \rightarrow y) \in C(x,y)$ can be easily calculated by a single $IFZ_{area}$ table lookup:

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} s' \rightarrow y) \\ &= IFZ^X_{area}(x, s', y) \end{aligned}$$

This table is managed by the current navigator, so that no additional intra-navigator network traffic is required for routing a vehicle within the destination area. For an adequate routing in case 2 travel times have to be accumulated across the different layers in BeeJamA.

Therefore, travel costs are composed of three parts:

- The travel times from the vehicle's current position at node $x$ to a border node of area $X$ (over neighbor $s \in S(x)$).
- The minimum travel times from a "suitable" border node of the current area $X$ to a border node of the destination area $Y$ on the net layer.
- The minimum travel times from a border node of area $Y$ to the destination node $y$.

The following cost paths are used to approximate these travel costs:

$$C(x,y) = \left\{\left(x \dot{\rightarrow} S(x) \rightarrow B(X)_i \overset{\min}{\rightarrow} B(Y)_j \overset{\min}{\rightarrow} y\right)\right\}$$

and

[2]Please note, that $\{(x \rightarrow \ldots \rightarrow M \rightarrow \ldots \rightarrow y)\} = \{(x \rightarrow \ldots \rightarrow m_1 \rightarrow \ldots \rightarrow y), \ldots, (x \rightarrow \ldots \rightarrow m_n \rightarrow \ldots \rightarrow y)\}$, if $M = \{m_1, \ldots, m_n\}$.

$$|C(x,y)| = |S(x)| \cdot |B(X)| \cdot |B(Y)_i|$$

The cost function is given by:

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} s \rightarrow B(X)_i) \\ &\quad + \check{c}(B(X)_i \rightarrow B(Y)_j) + \check{c}(B(Y)_j, y) \\ &= c(IFZ^X_{area}(x, n', B(X)_i)) \\ &\quad + \min(IFZ^{X,Y}_{net}(B(X)_i, B(Y)_j))) \\ &\quad + \min(IFZ^Y_{area}(B(Y)_j, y))) \end{aligned}$$

(Here the min notation is used to denote the minimum table entry from the first argument as current node and the second argument as destination). Only the last addend is unknown to the current navigator and the appropriate minimum value must be requested from the navigator of area $Y$.

For an adequate routing in case 3 travel times are composed of only two parts:

- The travel times from the vehicle's current position at node $x$ to a border node in its current area $X$.
- The minimum travel times from a "suitable" border node of the current area $X$ to the representative node $R_Y$ of area $Y$'s foraging region.

The key idea (adopted from the foraging behavior of bees in nature) is to route a vehicle in the rough direction towards the foraging region of its target area $Y$ until it gets close enough to its destination for it to be listed in the $IFZ_{net}$ table of a visited node.

The cost paths to approximate the travel costs are

$$C(x,y) = \left\{\left(x \dot{\rightarrow} S(x) \rightarrow B(X)_i \overset{\min}{\rightarrow} R_Y\right)\right\}$$

with

$$|C(x,y)| = |S(x)| \cdot |B(X)|$$

$$\begin{aligned} c(p) &= c(x \dot{\rightarrow} n' \rightarrow B(X)_i) + \check{c}(B(X)_i \rightarrow R_Y) \\ &= c(IFZ^X_{area}(x, n', B(X)_i) \\ &\quad + min(IFR^X_{net}(B(X)_i, R_Y)) \end{aligned}$$

Finally, to draw a next hop $s \in S(x)$ the aforementioned costs must be transformed into probabilities, so that higher costs (higher travel times) result in lower next hop probabilities. For a cost path $p_i \in \{p_1, \ldots, p_n\} \in C(x,y)$ the probability is given by

$$\begin{aligned} P(p_i): &= \frac{\left(\frac{c(p_i)}{\sum_{j=0}^{n} c(p_j)}\right)^{-1}}{\sum_{k=0}^{n}\left(\frac{c(p_k)}{\sum_{j=0}^{n} c(p_j)}\right)^{-1}} \\ &= \frac{1}{c(p_i)\sum_{k=0}^{n}\frac{1}{c(p_k)}}, \end{aligned}$$

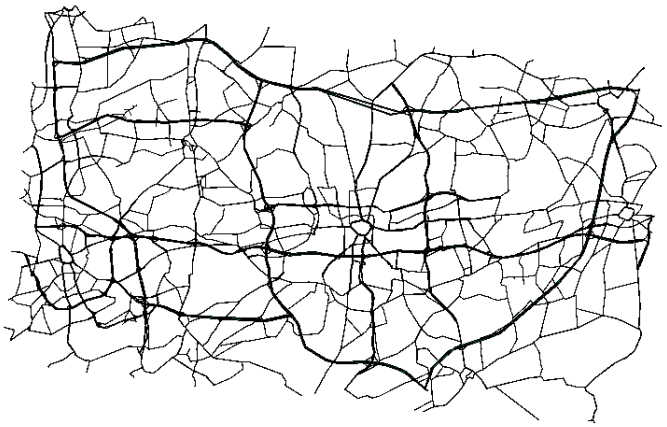


Fig. 6. Eastern Ruhrgebiet District

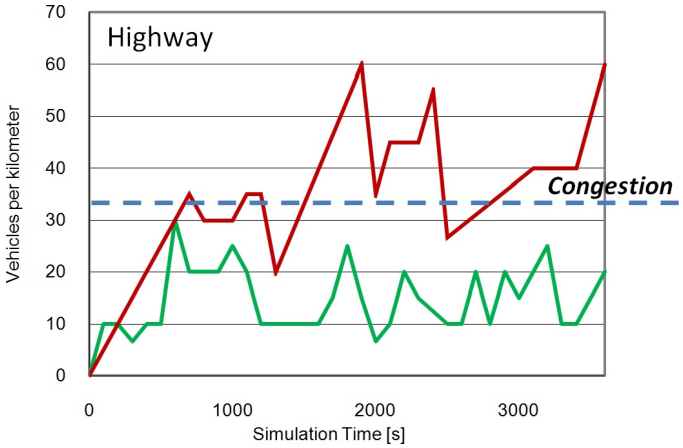



Fig. 7. Density

which is simply the normalization of the inverted normalized costs.

If the cost path $p_i = (x \dot{\rightarrow} s \rightarrow \ldots \rightarrow y)$ is chosen, then $s \in S(x)$ is the next hop sent back to the vehicle.

Bees in nature perform the waggle dance only if the quality of a discovered food source exceeds a certain threshold. Otherwise, the food source is discarded and no new foragers will be recruited. In analogy to this behavior, a route may only be selected if its travel cost does not exceed a dynamic threshold (and thus would avoid unacceptable detours).

## VI. Simulation Studies

For our simulation experiments on MATSim [6] we selected the major part of the Eastern Ruhr District (see Figure 6, bold lines indicate highways) in Germany, a densely populated, formerly heavy industry area, featuring a good of approx. 1850 sections and approx. 3800 intersections. We also selected a variety of characteristic real-world initial configurations and utilized our own graphical tools to follow the events as they occurred under BeeJamA and one of the commercially available shortest-path based services. We picked quite a few road sections that turned into bottlenecks under the shortest-path approach. This history for one section is to be observed in Figure 7. Here the car density is plotted over the simulation time. A threshold (see [10] for more details) is marked which indicates that any value above means congestion on this section. As it can be observed, the shortest-path approach (dark color) results in quite a far serious congestion while on the other hand BeeJamA (light color) leaves this section congestion-free. In another course of study - which we cannot display in detail, due to page limitations - we selected, starting again from real-world scenarios, a substantial subset of cars with fixed source and destination, and we kept track of their travel times in subsequent runs. Here again, it was clear that in the average over the runs, the travel times were kept considerably lower compared to shortest-path solutions.

## VII. Conclusions

The BeeJamA traffic guidance system is a highly flexible distributed multi-agent system. While self-organizing under the challenge of dynamic and very tight deadlines it makes sure that any directions are still accurate when carried out. No global information is needed for decision making. So altogether BeeJamA has the potential from the beginning to be robust under gradually growing degrees of penetration (among the drivers). Our results so far support this claim clearly. Also scalability is not a serious issue since the new structure over the navigation areas could be layered further in order to cover systems of the size of Germany. The idea behind is that the more distant a disturbance is the less influential it will be in a particular city, a principle that is borrowed from the swarm intelligence of honey bees.

The upcoming research will focus on more and more realistic scenarios and a much more detailed experimental insights. This will be subject of forthcoming publications.

## References

[1] W. Rothengatter. *External costs of transport, 2004,* http://www.uic.org/html/environnement/cd_external/pages/introduction.html*, last access at: 08/27/2010*

[2] M. Farooq. *Bee-Inspired Protocol Engineering - From Nature to Networks*. Springer, Berlin, 2009.

[3] T.D. Seeley. *The Wisdom of the Hive*. Harvard University Press, Cambridge, 1995.

[4] H. F. Wedde, S. Lehnhoff, S. Senge, and A. M. Lazarescu et. al. *A Novel Class of Multi-Agent Algorithms for Highly Dynamic Transport Planning Inspired by Honey Bee Behavior*. In: Proc. of the 12th IEEE Conf. on Emerging Technologies and Factory Automation, Patras, 2007.

[5] B. Tatomir, L.J.M. Rothkrantz, *H-ABC A scalable dynamic routing algorithm*, In: Recent Advances in Artificial Life, World Scientific Publishing Co. Pte. Ltd. 5 Toh Tuck Link, Singapore 596224

[6] G. Di Caro, M. Dorigo, *AntNet: Distributed Stigmergetic Control for Communications Networks*, In: Journal of Artificial Intelligence Research, vol. 9, pag. 317-365, 1998

[7] MATSim - Multi-Agent Transport Simulation Toolkit, official homepage at: http://www.matsim.org, last access at: 08/27/2010

[8] E. Bonabeau, M. Dorigo and G. Theraulaz. *Swarm Intelligence - From Natural to Artificial Systems*. Oxford University Press, Ney York, 1999.

[9] B.S. Kerner et al. *Traffic State Detection with Floating Car Data in Road Networks*. IEEE Intelligent Transportation Systems, 2005.

[10] L. Neubert et. al. *Statistical Analysis of Freeway Traffic*. In: Traffic and Granular Flow '99, Springer, 2000.

 ISBN: 978-1-61208-109-0 

# Laser Measurement System based maneuvering Target tracking formulated by Adaptive Competitive Neural Networks

Lokukaluge P. Perera
Centre for Marine Technology and Engineering
Technical University of Lisbon, Instituto Superior Técnico, Lisbon, Portugal
prasad.perera@mar.ist.utl.pt

Carlos Guedes Soares
Centre for Marine Technology and Engineering
Technical University of Lisbon, Instituto Superior Técnico, Lisbon, Portugal
guedess@mar.ist.utl.pt

***Abstract*—To improve safety and security issues, maneuvering target detection and tracking are important facilities for navigation systems. Therefore, conventional navigation systems are equipped with Radar-based systems for the same purpose. However, Radar systems suffer some practical problems that are associated with the targets in close quarter navigation. Furthermore, Radar singles attenuate with distance, weather (ie. rain) and sea conditions, where the target tacking performances are degraded. Therefore, a Laser Measurement System (LMS) is proposed in this study to overcome the problems faced by the conventional Radar systems at close quarter navigation as well as bad weather and environmental conditions. Furthermore, capabilities of a LMS to measure accurate distance in close proximity as well as to observe the shape and size of the target are illustrated. In this study, each target is approximated by a cluster of data points rather than a single point target that is the main contribution in this paper. The adaptive Neural Network approach is proposed as a method of tracking maneuvering targets that are represented by clusters of data points. Successful simulation and experimental results of target detection and tracking that are tested on a experimental platform, SICK© LMS, are also presented in this paper.**

***Keywords- Laser Measurement System, Competitive Neural Networks, Target tracking, Data Points Tracking***

## I. INTRODUCTION

Maneuvering target detection and tracking capabilities are important facilities for a navigation system to improve safety, security and survivability during its voyage. The conventional navigation systems are equipped with Radar-based systems to facilitate maneuvering targets and obstacles detection and tracking. However, Radar-based systems are suffered by practical problems especially with detection and tracking of targets in close quarter navigation. Furthermore, Radar singles attenuate with the distance, weather (ie. rain) and sea conditions, where the target tacking performances are degraded [1]. Therefore, under the distance, weather and environmental conditions, the frequent calibrations for Radar systems are required to improve its accuracy [2].

Furthermore, Radar-based systems are limited in evaluation of accurate range, bearing, shape and size of targets in long distance as well as close quarter navigation. The unsuccessful target detection and tracking in close quarter navigation could affect on inaccuracy of the distance measurements with respect to the targets and obstacles in the environment. Therefore, the errors in distance measurements can eventually affect on inaccurate collision risk evaluations and wrong navigational decisions.

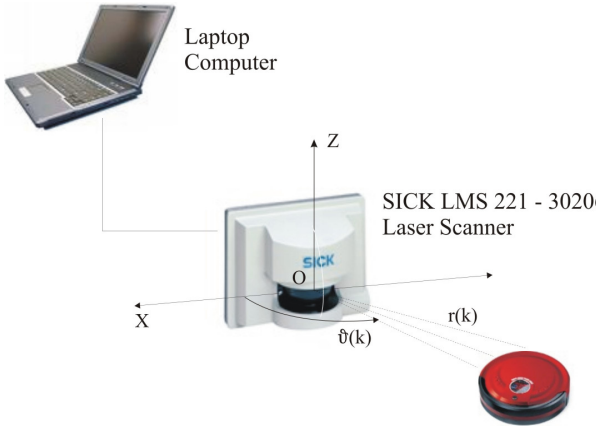



Figure 1. LMS Experimental Setup

This study proposes, a Laser Measurement System (LMS) that is integrated with an adaptive Neural Network algorithm for maneuvering target detection and tracking in close quarter navigation. Hence, these facilities can be formulated in navigation systems for accurate collision risk evaluations and better maneuvering decisions. The proposed LMS experimental platform in this study is presented in Figure 1. As presented in the figure, the experimental setup consists of a Laptop computer, where the adaptive Neural Network algorithm is implemented, SICK© LMS, which is the target detection sensor, and a moving target (ie. moving robot). Further details on this system are presented on Section V of this paper.

The work presented in this study is a part of ongoing effort to formulate an Intelligent Collision Avoidance System (ICAS) in ocean navigation as described in [3] and [4]. Therefore, two-dimensional target tracking formation with respect to ocean navigational conditions is considered in this study.

The organization of this paper as follows: An overview of recent developments in a LMS is presented in Section II. The proposed adaptive Neural Network approach is

 ISBN: 978-1-61208-109-0 

presented in Section III. The computational simulations are presented in Section IV. The experimental results generated by the LMS platform are presented in Section V. Finally, a brief conclusion is presented in Section VI of this paper.

## II. RECENT DEVELOPMENTS IN LMSS AND TARGET TRACKING

### A. Laser Measurement System (LMS)

People and vessel/vehicles detection and tracking are popular experimental applications in recent LMSs. The people tracking system, based on Laser range data and using a multi-hypothesis Leg-Tracker, in-cooperated with the Kalman filter with a constant velocity based model, is proposed by [5].

Almost all research and commercial applications in the LMS area are 2D due to the limitations in current LMS sensor technology. However, 3D LMS applications are also been proposed in some studies by circulating the sensor in a third dimensional axis. The 2D Laser-based obstacle motion tracking and predicting in a dynamic unconstrained environment using the Kalman filter [6], and the Particle Filters and Probabilistic Data Association [7] are presented in respective studies. The implementation of a LMS for detection and classification of 3D moving objects is illustrated by [8] and [9]. Furthermore, an experimental evaluation of a LMS for tacking people by a mobile platform is presented by [10] and classifications of people by a mobile robot is presented by [11].

The integration of a LMS and a vision system can use for successful obstacle tracking and classification due to its capabilities of capturing the comprehensive details of the targets as well as the environment. The combined Laser and vision based approach for simultaneous detection and tracking of multiple pedestrians based on the Bayesian method is proposed by [12].

Furthermore, several industrial applications of LMSs can be observed in recent literature: As a navigational aid for a truck-trailer combination vehicle system [13], a collisions warning system for a transit bus [14], a safe driving aid system for a car driving in polluted environment [15], an obstacle avoidance system for a car navigation system [16] and an obstacle detection system for an off-road vehicle [17] are presented in respective studies.

However, the most model based LMS and other sensor based target-tacking algorithms could not facilitate the dimensional based target tracking and a target is approximated to a single data point. Therefore, in this study, this concept is further elaborated to formulate a target as a cluster of data points during its tracking process, which is the main contributions in this study.

### B. Detection and Tracking of Moving Objects

Detection and Tracking of Moving Objects (DTMO) is one of the main research areas that was developed towards maneuvering target tracking. The main functionalities of the DTMO can be divided into three sections [18]: Scan unit, Target Classification unit and Target Tracking & Behavior Prediction unit.

The main objective of the Scan unit is to formulate geometrical clusters, where a cluster defined as a set of measured data points that could belong to a same object or multiple objects, of data points, lines and arcs with respect to targets and obstacles in the environment. However, this could be generated by the sensors (ie. Radar and LMSs) in the target tracking system.

The segmentation of data clusters by a geometrical method is proposed by [18]. Inscribed angle variations and recursive line-fitting methods for lines and arc/circles detection by LMS data are proposed by [19]. However, special considerations for the joints and break points should be considered during its segmentation of data clusters, in this method. Therefore, the proposed adaptive Neural Network approach can overcome the failures can occur in the segmentation process of data clusters due to varying geometrical constrains.

The main objective in the Target Classification unit is to formulate the Segment-Objects correspondence. The correspondence mainly classified into geometrical figures like circles or polygons. However, four classification methods are proposed by recent studies [20] for this prupose: Features to Features, Points to Features, Points to Points and Combinations. Furthermore, the identifications of geometrical figures and features are successful done by the Neural Network approach in recent studies [21].

Even though the proposed adaptive Neural Network approach is limited for detection and tracking of clusters of data points, this method can further develop for identification of geometrical figures and features of the targets. Finally, the Target Tracking and Behavior Prediction unit is proposed to estimate target current states and to predict future navigation trajectories. The EKF based system states estimation and maneuvering trajectory prediction for ocean vessel navigation is proposed in [22]. However, this area is beyond the scope of this paper.

## III. ADAPTIVE NEURAL NETWORK BASED DETECTION & TRACKING

### A. LMS Scan & Data Collection

The LMS experimental platform is presented in Figure 1. The LMS sensor generates respective range $\mathbf{r}(k) \in \mathbb{R}^R$ and bearing $\vartheta(k) \in \mathbb{R}^R$ values in polar coordinates as presented in the figure. The accumulated data clusters of range and bearing values that represent complete environmental conditions, including the stationary and moving targets, at the k-th time instant in polar coordinates can be written as:

$$\begin{aligned} \mathbf{r}(k) &= [r_1(k)\ r_2(k)\ \ldots\ r_R(k)] \\ \vartheta(k) &= [\vartheta_1(k)\ \vartheta_2(k)\ \ldots\ \vartheta_R(k)] \end{aligned} \tag{1}$$

Then the range and bearing values in polar coordinates are converted into Cartesian position coordinates. The i-th position data point in Cartesian coordinates $[^Rx_i(k)\ ^Ry_i(k)]$ can be formulated as:

$$
\begin{aligned}
{}^{R}x_i(k) &= r_i(k)\cos\left(\vartheta_i(k)\right) \\
{}^{R}y_i(k) &= r_i(k)\sin\left(\vartheta_i(k)\right)
\end{aligned}
\tag{2}
$$

Therefore, the i-th position data point of the data cluster at the k-th time instant can only have two coordinates of $[{}^{R}x_i(k)\ {}^{R}y_i(k)]$ that are measured by the sensor. However, these position data points should be normalized with respect to the maximum range of the sensor. The normalization requirements are further discussed in sub-section C of this main section. The normalization of the position coordinates can be written as:

$$
\begin{aligned}
x_i(k) &= \frac{{}^{R}x_i(k)}{R_{max}} \\
y_i(k) &= \frac{{}^{R}y_i(k)}{R_{max}}
\end{aligned}
\tag{3}
$$

where $R_{max}$ is the maximum range of the LMS sensor.

### B. Artificial Neural Networks

The theoretical foundation of artificial neurons is derived from biological concepts and theories in the brain and nervous system. An artificial neuron has several inputs that correspond to the synapses of a biological neuron. An artificial neuron has one output that is corresponding to the axon of a biological neuron. Each input of a neuron is corresponding to a certain weight value that influences the corresponding signal over the neuron output. This concept can formulate into a transfer function in an artificial neuron.

The transfer function calculates sum of the net input with respect to the assigned weight values and compares that with a certain threshold level to generate the neuron output [23]. The connection of several neurons in a combination of series and/or parallel formations can recognize as a Neural Network.

### C. Competitive Neural Network

The Competitive Neural Network (CNN) [24] integrated with an adaptive learning algorithm of the Instar Rule is proposed in this study for detection and tracking of maneuvering targets. The CNN is trained to track moving data clusters by competing its neurons, where a target is approximated for a cluster of data points. .

The structure of the CNN is presented in Figure 2. As presented in the figure, the CNN consists of four units: Scan unit (Data Points), Prototype vectors unit (**W**), Competition unit (**C**), and Feedback-loop (Instar Rule). The input to the CNN consists of a accumulated position vector $\mathbf{p}(k) \in \mathbb{R}^{Rx3}$. The prototypes vectors, $\mathbf{W}(k) \in \mathbb{R}^{Sx3}$, are stored as rows vectors in section **W,** that are target tracking neurons of the CNN. The net input $\mathbf{n}(k) \in \mathbb{R}^{R}$, is the input to the Competition unit, **C**, and $\mathbf{a}(k) \in \mathbb{R}^{S}$, is the output from the Competition unit, **C**, at the k-th time instant. Finally, the feedback loop, associated with the Instar Rule that is proposed to adjust the prototype neurons to continue tracking of maneuvering targets.

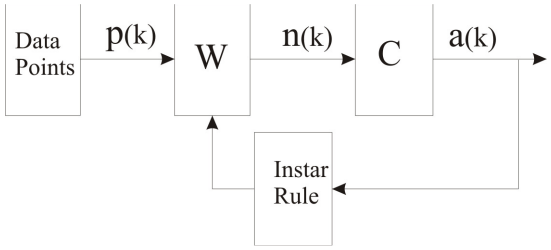



Figure 2. Competitive Neural Network

#### 1) Competitive Layers

The Competition unit, **C,** consists of a transfer function that is used to generate competition among neurons. Hence, the proposed transfer function can be written as:

$$
\begin{aligned}
\mathbf{a}(k) &= \text{compet}\left(\mathbf{n}(k)\right) \\
&= \text{compet}\left(\mathbf{W}(k)\,\mathbf{p}(k)\right)
\end{aligned}
\tag{4}
$$

where the competitive (compet) transfer function can be further elaborated as:

$$
\text{compet}\left(\mathbf{n}(k)\right) = \begin{cases} 1 \text{ for neron with } \max \mathbf{n}(k) \\ 0 \qquad \text{all other nerons} \end{cases}
\tag{5}
$$

and the accumulated position vector $\mathbf{p}(k) = [\mathbf{p}_1(k)\ \mathbf{p}_2(k)\ \ldots\ \mathbf{p}_R(k)]$ is associated with the i-th position vector, $\mathbf{p}_i(k) = [x_i(k)\ y_i(k)\ z_i(k)]$ that represents the position of x, y and z coordinates of a data cluster as described previously. However, only normalized x and y position coordinates are calculated from equation (3). For a fair competition among neurons, each position vector, $\mathbf{p}_i(k) \in \mathbb{R}^3$, should have a unit magnitude condition. Hence, the position value of $z_i(k)$ can be derived considering a unit magnitude condition as proposed previously and can be formulated as:

$$
|\,\mathbf{p}_i(k)\,| = \sqrt{x_i^{\,2}(k) + y_i^{\,2}(k) + z_i^{\,2}(k)} = 1
\tag{6}
$$

Hence the coordinate $z_i(k)$ can be calculated considering equation (6) that gives a unit magnitude condition for each data point in the data culster. The coordinate $z_i(k)$ can be calculated as:

$$
z_i(k) = \sqrt{1 - x_i^{\,2}(k) - y_i^{\,2}(k)}
\tag{7}
$$

One should note that this implementation can interpret as a transformation of 2D space position coordinates in the sensor range into 3D space position coordinates with a unit magnitude condition. Therefore, initially x and y coordinates are normalized considering the sensor maximum range (see equation (3)) that is an essential requirement of the neural competition.

 ISBN: 978-1-61208-109-0 

Furthermore, the net input, $\mathbf{n}(k)$, can calculate from the scalar product between two vectors $\mathbf{W}(k)$ and $\mathbf{p}_i(k)$ as presented in equation (4). This scalar product between two position vectors related to the distance between position vector, $\mathbf{p}_i(k)$, and each prototype vectors $\mathbf{w}_j(k)$, where $\mathbf{W}(k) = [\mathbf{w}_1(k)\ \mathbf{w}_2(k)\ \ldots\ \mathbf{w}_S(k)]$. A unit magnitude condition for each prototype vector, $\mathbf{w}_j(k)$, should also be considered for fair competition among neurons. Hence the j-th prototype vector, $\mathbf{w}_j(k)$, magnitude condition can be written as:

$$\| \mathbf{w}_j(k) \| = 1 \tag{8}$$

In the Competition unit, **C,** (see Figure 2), the distance between position vector, $\mathbf{p}_i(k)$ to each prototype vector $\mathbf{w}_j(k)$ is calculated. This concept can further be elaborated as:

$$\mathbf{n}_i(k) = \mathbf{W}(k)\mathbf{p}_i(k) = \begin{bmatrix} \mathbf{w}_1^T(k) \\ \mathbf{w}_2^T(k) \\ \vdots \\ \mathbf{w}_S^T(k) \end{bmatrix} \mathbf{p}_i(k) = \begin{bmatrix} \mathbf{w}_1^T(k)\mathbf{p}_i(k) \\ \mathbf{w}_2^T(k)\mathbf{p}_i(k) \\ \vdots \\ \mathbf{w}_S^T(k)\mathbf{p}_i(k) \end{bmatrix} = \begin{bmatrix} \cos\theta_1(k) \\ \cos\theta_2(k) \\ \vdots \\ \cos\theta_S(k) \end{bmatrix} \tag{9}$$

The i-th net input of scalar product between two vectors, $\mathbf{n}_i(k)$, is equal to $\cos(\theta_i(k))$, where $\theta_i(k)$ is the angle between a position vector, $\mathbf{p}_i(k)$, and a prototype vector, $\mathbf{w}_j(k)$. However, the scalar product between two vectors, $\mathbf{n}_i(k)$, is the input to the competitive transfer function. Therefore, the neuron, whose prototype vector is in the direction closest to the respective position vector, $\mathbf{p}_i(k)$, is assigned output of 1 and others are assigned 0 by the transfer function as formulated in equation (5).

This concept can further elaborate as a situation where the closet neuron gets excited by a data cluster and the excited neuron takes over all data points in the respective data cluster. However, after winning the data cluster, the prototype vector of the respective neuron should be improved (should move further closer to the data cluster).

This continues process consists of two different iteration loops. The first iteration loop formulates a continuous mechanism, where the wining neuron continuously gets closer to its respective data cluster. The second iteration loop formulates another continuous mechanism, where the dynamic data clusters that are observed by the sensor at different time instants are introduced into the CNN. Therefore, a capable learning rule should be formulated to facilitate proper update of the winning neurons with respect to different data cluster conditions.

*2) Competitive Learning*

Initially, the values of prototype vectors, $\mathbf{W}(k)$, in the CNN, are assumed to be unknown. Therefore, the learning rule is expected to calculate appropriate values for the prototype vectors. This concept is categorized as unsupervised learning. When a competitive layer excites a neuron that is closest to the data cluster, then the learning rule will use to modify the appropriate prototype vectors in the CNN to move close to the data cluster in this process. The Instar Rule is proposed in this study as an unsupervised learning mechanism to modify the appropriate prototype vectors in the CNN.

*3) Instar Rule*

The Instar Rule that is derived from the Hebb Rule is illustrated in [24] is briefly discussed in this section. The unsupervised Hebb Rule to update prototype vectors can be written as:

$$\mathbf{W}(k) = \mathbf{W}(k-1) + \alpha\mathbf{a}(k)\mathbf{p}^T(k) \tag{10}$$

where $\alpha$ is a learning rate. However, a constant learning rate could be a disadvantage in the learning process of a neural network, where it could affect on the error convergence rate. Even though the beginning of a learning process a higher learning rate is an advantage to the neural network, with the error reduction it could be a disadvantage. Hence, to improve the Hebb Rule, a weight decaying term that is proportional to $\mathbf{a}_i(k)$ and $\mathbf{W}(k\text{-}1)$ is introduced. Equation (10) with a weight decaying term can be written as:

$$\mathbf{W}(k) = \mathbf{W}(k-1) + \alpha\mathbf{a}(k)\mathbf{p}^T(k) - \gamma\mathbf{a}(k)\mathbf{W}(k-1) \tag{11}$$

where $\gamma$ is the decay rate. Furthermore, assuming $\gamma = \alpha$, equation (11) can be written as:

$$\mathbf{W}(k) = \mathbf{W}(k-1) + \alpha\mathbf{a}(k)\left(\mathbf{p}^T(k) - \mathbf{W}(k-1)\right) \tag{12}$$

Equation (12) is called as the Instar Rule that is proposed as an unsupervised learning rule for the CNN in this study.

## IV. COMPUTATIONAL IMPLEMENTATION AND SIMULATIONS

The computational simulation of a multi-target tracking situation is presented in Figure 3. The simulation consists of two moving targets that are presented by two clusters of data points. Furthermore, two prototype vectors are also assigned in this simulation to tack both data clusters.

 ISBN: 978-1-61208-109-0 

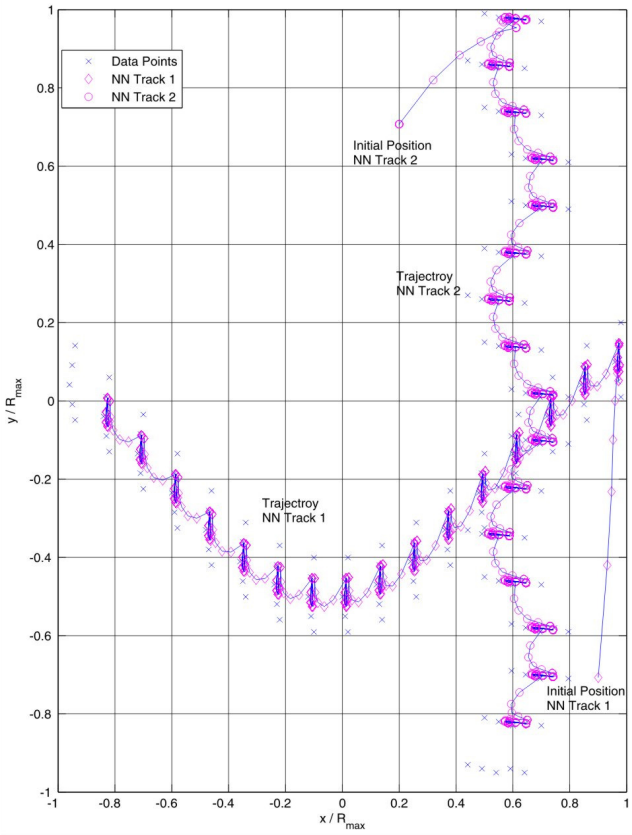



Figure 3. Computational Simulations : Multi-Target Tracking

The target tracking algorithm, simulated in this study, consists of two main loops: the LMS target scanning loop and the CNN target tracking loop. The main objective of the LMS target scanning loop is to scan the environment and to observe the stationary and moving targets as an accumulated data cluster. Then this information will transfer into the CNN target tacking loop. The main objective of the CNN target tacking loop is to adapt the CNN to track target maneuvers by updating its respective prototype vectors. This should be done by the proposed learning rule.

The two prototype vectors of the CNN are called as NN (Neural Network) Tracks 1 and 2. As presented in the figure, the NN Tacks 1 and 2 are presented by ◊ and ○ respectively. The initial prototype vector values of the NN Tracks 1 and 2 can be any arbitrary values as presented in the initial positions of the NN Tacks 1 and 2 (see Figure 3). Finally, both NN Tracks are adapted its prototype vectors to track moving targets that represented by two clusters of data points. The tracking trajectories of the neurons are presented by Trajectory NN Tracks 1 and 2 in the figure.

Furthermore, it is observed that NN Tracks 1 and 2 are finally converged into approximate mean values of the respective data clusters at each time instant. Therefore, the mean position values can be considered as measurement positions of the targets at each time instant and that can be used for further analysis of target state estimations and trajectory predictions [22].

## V. EXPERIMENTAL PLATFORM AND SIMULATION RESULTS

### A. *Laser Measurement System*

The experimental platform is presented in Figure 1. As presented in the figure, the hardware section mainly consists of SICK© Laser Measurement System (LMS). The SICK© LMS is an active position measurement unit that operates by measuring the time of flight of Laser light pulses, where Laser beam pulses are emitted by the sensor and reflected due to the objects in the environment [25]. However, the LMS is designed to scan 2D space and to collect range and bearing data of the targets that are located in the environment.

The SICK© LMS model of LMS221, that is designed for marine environment is used in this study. This sensor is capable of scanning bearing angle of $180^0$ with $0.5^0$ accuracy field views with 75 Hz scanning frequency. The operating range of 8 m with the minimum linear and angular resolution of 1 mm and $0.25^0$ are initially programmed into the sensor. The SICK© LMS data communication is facilitated by RS-232 with the speed of 9.6 kBd.

Furthermore, the experimental platform consists of a Laptop computer with Windows© operating system and a power supply unit to power the LMS sensor. The Laptop computer is equipped with the RS-232 connection to communicate with the LMS sensor.

### B. *Software Architecture*

The software architecture that is used in this study mainly consists of LABVIEW© Real-time platform. Further MATLAB© toolbox of neural networks is also integrated into the LABVIEW© Real-time platform for implementation of the CNN.

### C. *Experimental Results*

The experimental result of a stationary and moving target tracking situation in Real-time environment is presented in Figure 4. The measurements are noted in mille-meters (mm) of SI units in the figure. The two targets, a stationary target and a moving target, are considered in this experiment. The stationary target is located in the middle of the figure and the moving target is circulating around the stationary target as presented in the figure. The moving target is presented by a moving cluster of data points. The CNN consists of two NN Tracks to track both targets as presented in the figure. Furthermore, the stationary target is monitored by the NN Track ○ and moving target is monitored by the NN Track ◊ are also presented in the figure.

However, the CNN tracking region is limited by upper, lower, left and right boundary values 9000 (mm), 10 (mm), -1200 (mm) and 1200 (mm), respectively. As presented in the figure, NN Track ◊ is following each point in the data cluster of the moving target alone its maneuvering trajectory. As a conclusion, the experimental results have shown that the NN Track ○ and Track ◊ are successfully tracking both stationary and moving targets as observed in the simulations.

 ISBN: 978-1-61208-109-0 

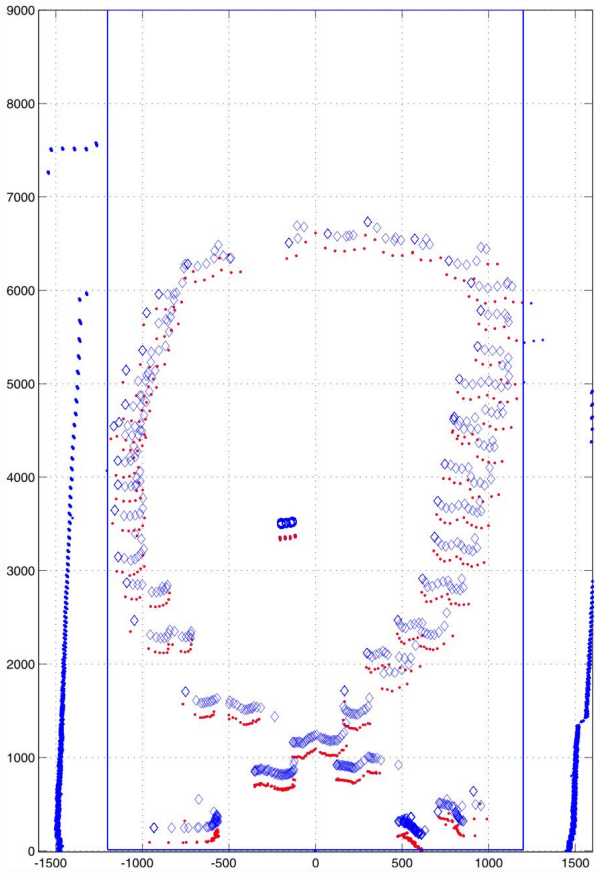



Figure 4. Experimental Results: Stationary and Moving Target Tracking

However, the data points beyond the limits of upper, lower, left and right boundary values are ignored in this analysis. These data points are located beyond the simulation limits, which represent other stationary and moving objects in the experimental environment.

## VI. CONCLUSION

The dimension based target detection and tacking are main contributions in this study, where the most of target tracking methods are simulated for a data point or an approximated small data cluster based targets. Furthermore, one of the popular machine learning applications of an adaptive Neural Network associated with an unsupervised learning algorithm, the CNN, is implemented and successful simulation and experimental results are obtained in this study.

Even though the Neural Network applications are extensively used for recognition of stationary data patterns, moving data clusters can also be detected and tracked by the proposed method. Even though, the proposed CNN behave as an effective adaptive network for tracking targets, it also been affected by some inherited problems.

The first, the selection of a learning rate should be compromised with the target tracking speed. However, this compromise could affect on the stability of the prototype vectors.

The second, the stationary and moving target tracking under complex environmental conditions: several neurons can track different parts of the same target and one neuron can track several targets in close range navigation. This is another challenge that is faced in this CNN approach. However, this situation can be solved by selecting proper number of neurons with respect to the targets in the environment.

. Furthermore, the Laser-based CNN approach can further develop for identification and classification of maneuvering targets where the Neural Network approach is extensive implemented on statistical pattern recognition [26]. Furthermore, integration of image based (ie. Infra-red) facilities could improve the target detection and tracking process [27]. Hence, the integration of illustrated features (ie. identification and classification) into target detection and tracking are proposed as future work in this study.

## ACKNOWLEDGMENT

This work has been made within the project ”Methodology for ships maneuverability tests with self-propelled models”, which is being funded by the Portuguese Foundation for Science and Technology (Fundação para a Ciência e Tecnologia) under contract PTDC/TRA/74332 /2006. The research work of the first author has been supported by a Doctoral Fellowship of the Portuguese Foundation for Science and Technology (Fundação para a Ciência e Tecnologia) under contract SFRH/BD/46270/2008.

## REFERENCES

[1] S. E. Giangrande and A. V. Ryzhkov, “Calibration of dual-polarization radar in the presence of partial beam blockage”, *Journal of Atmospheric and Oceanic Technology,* vol. 22, pp. 1156–1166, 2004.

[2] R. Naranjo, “Radar revisited”, Ocean Navigator Online, http://www.oceannavigator.com/, [retrieved: August, 2010].

[3] L. P. Perera, J. P. Carvalho, and C. Guedes Soares, “Decision making system for the collision avoidance of marine vessel navigation based on COLREG rules and regulations,” in *Proceedings of 13th Congress of International Maritime Association of Mediterranean*, Istanbul, Turkey, 2009, pp. 1121–1128.

[4] L. P. Perera, J. P. Carvalho, and C. Guedes Soares, “Smooth transition between fuzzy regions to overcome failures in fuzzy membership functions of decisions in collision avoidance of ocean navigation,” in *Proceedings of 25th Mini-EURO Conference on Uncertainty and Robustness in Planning and Decision Making*, Coimbra, Portugal, 2010, pp 1-8.

[5] K. O. Arras, S. Grzonka, M. Luber, and W. Burgard, “Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities,” in *Proceedings of the 2008 IEEE International Conference on Robotics and Automations*, CA, USA, 2008, pp. 1710–1715.

[6] M. Berker, R. Hall, S. Kolski, K. Macek, R. Siegwart, and B. Jensen, “2d laser-based probabilistic motion tracking in urban-like environments,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 31, no. 2, pp. 83–96, 2009.

 ISBN: 978-1-61208-109-0 

[7] A. Almeida, J. Almeida, and R. Araujo, "Real-time tracking of multiple moving objects using particle filters and probabilistic data association," *Automatika*, vol. 46, no. 1-2, pp. 39–48, 2005.

[8] A. Lourenco, P. Freitas, M. I. Ribeiro, and J. S. Marques, "Detection and classification of 3d moving objects," in *Proceedings of the 10th Mediterranean Conference on Control and Automation*, Lisboa, Portugal,, 2002.

[9] G. Gallagher, S. Srinivasa, and J. Andrew, "GATMO : A generalized approach to tracking movable objects," in *IEEE International Conference on Robotics and Automations*, 2009, pp 2043-2048.

[10] M. Linstrom and J. O. Eklundh, "Detection and tracking moving objects from a mobile platform using a laser range scanner," in *Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hawaii, USA, 2001, pp. 1364–1369.

[11] M. Luber, K. O. Arras, C. Plagemann, and W. Burgard, "Classifying dynamic objects : An unsupervised learning approach," in *Proceedings of the Robotics: Science and Systems IV*, Zurich, Switzerland, 2008, pp 270-277.

[12] X. Song, J. Chi, H. Zhao, and H. Zha, "A bayesian approach : Fusion of laser and vision for multiple pedestrians tracking," *International Journal of Advanced Computer Engineering*, vol. 3, no. 1, pp. 1–9, 2008.

[13] R. Stahn, G. Heiserich, and A. Stopp, "Laser scanner-based navigation for commercial vehicles," in *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, Istanbul, Turkey, 2007, pp. 969–974.

[14] R. A. Maclachlan and C. Mertz, "Tracking of moving objects from a moving vehicle using a scanning laser rangefinder," in *Proceeding of the IEEE Intelligent Transportation Systems Conference*, Toronto, Canada, 2006, pp. 301–306.

[15] H. Hirose, K. Katabira, H. Zhao, and R. Shibasaki, "A study for safe driving using a laser scanner integration of the sensor on motionless objects and moving objects," in *Proceedings of the Asian Association on Remote Sensing*, Colombo, Sri Lanka, 2008.

[16] T. C. Ng, J. I. Ibanez-guzman, J. Shen, and Z. Gong, "Vehicle following with obstacle avoidance capabilities in natural environments," in *Proceedings of International Conference on Robotics and Automation*, 2004, pp. 4283–4288.

[17] C. S. Dima, N. Vandapel, and M. Hebert, "Sensor and classifier fusion for outdoor obstacle detection: An application of data fusion to autonomous off-road navigation," in *Proceedings of the 32nd Applied Imagery Pattern Recognition Workshop*, 2003, pp. 255–262.

[18] A. Mendes, L. C. Bento, and U. Nunes, "Multi-target detection and tracking with a lasers canner," in *2004 IEEE Intelligent Vehicles Symposium*, Parma, Italy, 2004, pp. 796–800.

[19] J. Xavier, M. Pacheco, D. Castro, A. Ruano, and U. Nunes, "Fast line arc/circle and leg detection from laser scan data in a player driver," in *Robotics and Automation, 2005, Proceedings of the 2005 IEEE International Conference on*, 2005, pp. 3930–3935.

[20] C. Wang and C. Thorpe, "Simultaneous localization with detection and tracking of moving objects," in *IEEE int. Conf. on Robotics and Automation*, Washington DC, 2002, pp. 2918–2924.

[21] Z. Jie-yu, "A novel recurrent neura network for face recognition," *Journal of Software*, vol. 12, no. 8, pp. 1128–1139, 2001.

[22] L. P. Perera and C. Guedes Soares, "Ocean vessel trajectory estimation and prediction based on extended kalman filter," in *Proc. 2nd International Conferrence on Adaptive and Self-adaptive Systems and Applications*, Lisbon, Portugal, 2010, (In Print).

[23] M. Cirstea, A. Dinu, J. Khor, and M. McCormick, Eds., *Neural and Fuzzy Logic Control of Drives and Power Systems*, 1st ed. MA, USA: Elsevier Science, 2002.

[24] M. T. Hagan, H. B. Demuth, and M. H. Beale, Eds., *Neural Network Design*. Boston: PWS Publishing, 1996.

[25] Technical Description, "LMS200/211/221/291 laser measurement systems." , http://www.sick.com, [retrieved: August, 2010].

[26] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, UK: Clarendon Press, 1995.

[27] X. Jiping, I. U. Haq, C. Jie, D. Lihua and L. Zaiwen, "Moving target detection and tracking in FLIR image sequences based on thermal target modeling," in *Proceedings of the International Conference on Measuring Technology and Mechatronics Automation*, 2010, pp. 715–720.

 ISBN: 978-1-61208-109-0 

# Adaptive Immunity through Differential Elasticity

Fatma Mili
Computer Science and Engineering
School of Engineering and Computer Science
Rochester, Michigan, US
e-mail: mili@oakland.edu

Nancy Alrajei
Computer Science and Engineering
School of Engineering and Computer Science
Rochester, Michigan, US
nmalraje@oakland.edu

***Abstract*— Malicious attacks are often targeted to affect the most vulnerable or most critical resources of a system. In sensor networks, because of the large amount of inherent redundancy, the most serious threats are the ones attacking critical paths in the network attempting to break them thus disrupting the overall function of the network. In this paper we define a set of graph properties that characterize the level of vulnerability of specific links. We use these properties to define a bio-inspired model of self-organization and adaptive reorganization that impart networks with resilience in the face of a variety of scenarios from simple power depletion to targeted malicious attacks.**

***Keywords- fault-tolerance; managing redundancy; k-connectedness; differential k-connectedness; elasticity.***

## I. INTRODUCTION

Redundancy has always been the key ingredient used to provide fault tolerance [10]. Sensor networks are no exception [1]. In fact, they are by design redundant since they consist of a set of interchangeable, functionally overlapping sensor nodes. Upon deployment, a sensor network is generally highly redundant through the use of more nodes than strictly necessary. Its level of redundancy decreases as nodes fail, get depleted of their power [6], or fall victims of unintentional or malicious attacks. An important issue when deploying a sensor network is thus in determining the initial density required that will ensure that the network would remain alive throughout the lifetime of the application (from weeks to months or longer) [9]. When the major risk is depletion and accidental failure, it is reasonable to expect that uniformly deploying more nodes is a quasi optimal way to increase the lifetime of the network. When the risk is malicious attacks, on the other hand, just adding more nodes becomes a very weak strategy because node failures are not random. A malicious attacker can choose which nodes to attack and can concentrate efforts at breaking the network by targeting specific areas. In this paper we discuss the issue of using redundancy and managing redundancy in a way to maximize fault-tolerance. The question we address is: "Suppose that I am willing to devote h times more nodes than needed. What is the best use of the additional nodes? More specifically, given an area A of interest, given a set M of nodes, (1) What is the optimal organization for these nodes? (2) How can we get the nodes to self-organize in a way that would be a good approximation of this optimal organization? (3) How can we get the nodes to autonomously keep adapting their organization to faulty nodes in the network?"

This paper is organized as follows: In Section II, we discuss regular patterns of redundancy and metrics used to capture levels of fault-tolerance. We conclude that these regular patterns and associated metrics are inadequate when it comes to increasing tolerance against malicious attacks. In Section III, we propose an alternative organization and associated metrics that are also based on redundancy, but the redundancy is not uniform but concentrated around areas of the network that are either more vulnerable or whose failure is more consequential. In Section IV, we propose distributed algorithms run on the individual nodes that will make the network organize more strategically and reorganize to reflect node failures. We summarize and conclude in Section V.

## II. TRADITIONAL MEASURES OF REDUNDANCY

We consider a sensor network used to monitor some parameters (e.g., detect human/animal presence) over an area (assume a square) A of interest. We further assume that all nodes are identical with a sensing range radius $R_s$ and a communication range $R_c$. There are typically two concerns: Coverage and Connectivity. Coverage refers to the fact that every point of the area A is within sensing range (distance no larger than $R_s$) of some node. Connectivity refers to the fact that between any pair of nodes (n, n') there is a sequence $n_0$=n, $n_1$, $n_2$, … $n_k$=n' such that the distance between any $n_i$, $n_{i+1}$ is no longer than the communication range $R_c$; in other words, any two nodes are able to communicate with each other. The network created by the set of nodes defines a graph whereby the nodes are the vertices of the graph and there is an edge between any two nodes that are within communication range of each other. When coverage is the only concern, the minimal number of nodes required to cover area A is proportional to the ratio $L^2/ R_s$ where L is the length of the sides of the square area. This minimal number of nodes can be obtained by organizing the nodes in a lattice of equilateral triangles of side $\sqrt{3}\ R_s$ as is shown in Figure 1. When we are concerned with both connectivity and coverage, the optimal configuration depends on the relationship between $R_s$ and $R_c$. If the communication range is equal to or larger than $\sqrt{3}\ R_s$, the optimal configuration for coverage more than satisfies connectivity [4,9]. The lattice, in fact will ensure that every node is at the center of a

 ISBN: 978-1-61208-109-0 

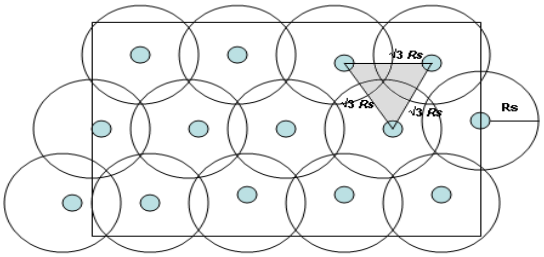



Figure 1. Optimal organization for coverage.

hexagon, and thus is connected to the six corners of that hexagon thus providing a high level of connectivity. In many applications, notably those where the phenomenon being monitored is continuous and not discrete, $R_c$ becomes the most critical parameter and connectivity the most critical issue. This is the subject of our focus here.

To ensure connectivity, when the sensing range is large enough, it is sufficient to organize the nodes in a chain that meanders through the surface. Resilience and fault tolerance require more than simple coverage and connectivity. We need connectivity even after nodes fail, exhaust their power, or get corrupted. In other words, we need to ensure that nodes are connected in more than one way so that when some of these ways fail others remain available. This concept of multiple and redundant connectivity has been identified in the literature under the name of k-connectivity and formally defined below.

The vertex connectivity $K_v(G)$ of a connected graph G is the minimum number of vertices whose removal can either disconnect G or reduce it to a 1-vertex graph.

A graph G is k-vertex-connected if G is connected and Kv(G) >= k. If G has non-adjacent vertices, then G is k-connected if every vertex-cut has at least k vertices.

The edge connectivity $K_e(G)$ of a connected graph G is the minimum number of edges whose removal can disconnect G. A graph G is k-edge-connected if G is connected and $K_e(G)>=k$. G is k-edge-connected if every edge cut has at least k edges. In the case of sensor networks, failures happen at the vertex level rather than at the edge level, thus we are interested in k-vertex-connectivity rather than k-edge-connectivity. In the reminder of this paper we will simply talk about k-connectivity to refer to k-vertex-connectivity. In a k-connected graph, between any two nodes there are at least k disjoint paths [14]. In a k-connected graph, every cut has at least k vertices.

The concept of k-connectivity has been introduced in graph theory and predates the widespread of sensor networks. With the emergence of sensor networks and their applications, many researchers identified the potential of this property for fault tolerance and fault repair [2]. One key challenge in organizing a sensor network so as to obtain the highest connectivity with the minimal number of nodes is that most of these problems are NP hard [5]. For this reason, many researchers in sensor networks, in addition to resorting to simplifying, but application-adequate models [2], focus on finding linear time approximation algorithms. Bredin *et al.* [5] investigated heuristic algorithms for deciding which nodes to wake when some nodes fail and for identifying locations where additional nodes should be placed. Wu and Li [12] and Alzoubi *et al.* [2] focus on the connectivity, not of the whole network, but of the subset of nodes that play a key role in routing. Specifically, they propose approximate algorithms for k-connected m-dominating sets. In [4], Bai *et al.* calculate the optimal deployment for the purpose of achieving k-connectivity. They discuss various patterns such as the triangular lattice and square grids. In [13], Xing *et al.* discuss protocols for ensuring multiple coverage and connectivity. In particular, by using a communication range twice as large as the sensing range, they distribute the nodes so as to ensure coverage, and by the same token have a high level of connectivity. In [8], Khelifa *et al.* discuss the inadequacy of traditional k-connectivity as a true measure of the level of redundancy and level of fault tolerance of a network, notably for large networks. In effect, as the network grows in size, the probability that the k nodes that fail belong to the same cut becomes very small. Conditional connectivity measures the level of fault tolerance in a network by focusing on the probability of failure of all the nodes of the same cut. While Khalifa *et al.* focus on the quantitative aspects, we address here the pragmatic aspects of organizing and reorganizing the network to maximize fault-tolerance. The approach proposed in this paper shares the underlying premise of Ammari and Das [3] by accounting for the conditional probability that the failure of a node will indeed lead to a failure of the network. We motivate our approach by first introducing three intuitive concepts.

Given a square surface area of side length L, and given a set of nodes N, what configuration would statistically maximize the length of time that the network is connected? The rationale here is that given a value k (e.g., 5), it may not be the best strategy to strengthen connectivity in a uniform way as not all vertices are equally useful and not all vertices are equally vulnerable. We discuss multiple considerations when considering the importance and vulnerability of the vertices.

The level of redundancy of a node. In a graph that is 1-connected, there is no redundancy. Every node failure can disconnect the graph. In a k-connected graph, the k-1 first failures are safe because every vertex-cut has at least k vertices. When all cuts have exactly k vertices, and if we call p the probability that any node fails within a time period P, the probability that the network gets disconnected within that same time period from that specific cut is $p^k$. In practice, different cuts have different sizes. The larger is the size of the cut the lower is the *conditional probability* that the network gets disconnected when a node of that cut fails. Nodes that belong to large cuts have a lower conditional probability of causing the network to fail when they fail. They are therefore "less critical" or "lower impact" (more

disposable) than nodes that belong to smaller cuts. In Figure 2, if we only focus on the two cuts shown, node a is less critical than nodes d and e. Node a belongs to a cut with three nodes, should it fail the connectivity will be ensured through c and e whereas a failure of d makes the network connectivity dependant on only one node: e.

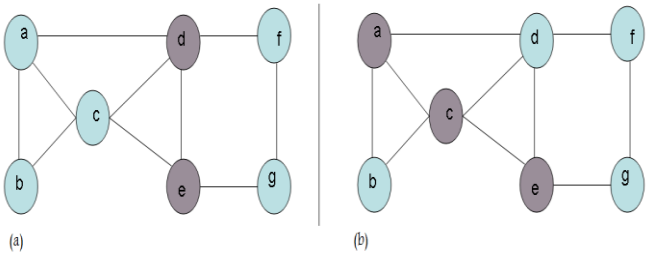



Figure 2. In (a) cut size is 2; in (b) cut size is 3.

Centrality of a node: Chipping vs. chattering the network. Given a connected graph, given a cut that divides the graph in connected components, the number and relative sizes of the resulting connected components are important in determining the impact of such a cut. A cut that barely "chips" the network by isolating a very small connected component from the rest of the network is less damaging than a cut that breaks it apart in many pieces or in two large pieces. The type of breakage that results from a cut characterizes the impact of losing all the nodes of a cut. Nodes in a cut that barely chips are less critical than nodes in cuts that chatter the network. This is illustrated in Figure 3. Losing node d results in disconnecting g but leaving the rest of the network together. By contrast, losing node c breaks the network into 4 components.

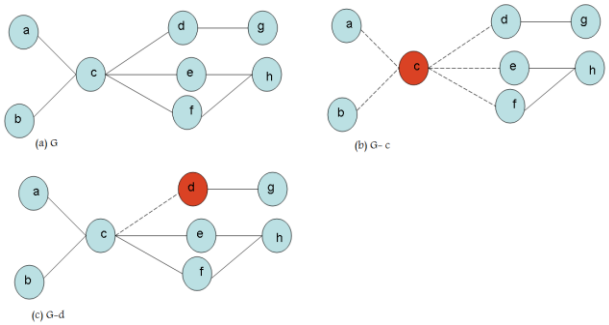



Figure 3. (a) Connected Graph G, (b) G after failure of c, (c) G after failure of d.

"Diameter" of a node: the intensity of the flow that goes through it. The network's functionality is captured by the flow circulating through it. Underlying every non directed sensor network, there is in fact a directed network where most communication takes place from and towards the base station. For example, in the graph in Figure 4, even though all nodes have similar connectivity degrees with communication flowing in both directions, there is an (many) underlying tree showing the routing of information from the base station to the rest of the network and from the rest of the network towards the base station. Thus, what is even more important than the connectedness of a node is the amount of information (flow) that that node is responsible for. For example, the flow of the incoming information (blue routing tree) is shown in the graph where the number associated with every node represents the number of measurements that node transports (effectively, it is the size of the routing sub-tree rooted at that node).

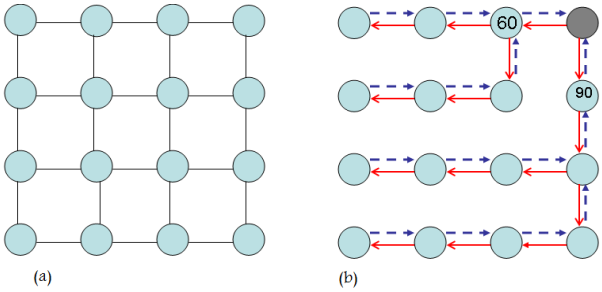



Figure 4. Flow through the network

Before we discuss these 3 concepts further we have to recognize that the first two concepts characterize a node based on "the" cut under consideration. Generally, a node belongs more than one cut, therefore a full characterization of a node would need to combine all cuts. We will not do that.

These concepts are presented here simply as an intuitive motivation for the following section. We are not planning on quantifying them or computing them. What we discuss here instead is the way in which they are inter-related. The level of redundancy of a node is the redundancy that is formally captured by the concept of k-connectivity except that k-connectivity sets an *initial* lower bound and captures the minimal level of redundancy present *in the network*. The conditional probability that the network gets disconnected when a node fails on the other hand, is associated with a node is a local measure of how much back up a node has. The centrality and diameter of a node are related in the sense that the flow that goes through a node is typically the flow from the sub-tree defined by that node towards the base station. The smaller is the sub-tree, the more the failure is likely to result on a chipping; the larger is the sub-tree, the more the failure is likely to result in a chattering. Therefore, both concepts correlate. Measuring one or the other will do. The key is to organize the network so that nodes with a high flow have a high level of redundancy.

## III. DIFFERENTIAL REDUNDANCY

### A. *Differential Connectivity*

The idea behind differential connectivity is that given a set of nodes available, instead of placing the nodes so as to ensure a uniform size cut everywhere throughout the network, and thus decreasing the conditional probability of

 ISBN: 978-1-61208-109-0 

failure of the network if a node fails, we should instead place the nodes so as to decrease the probability proportionally to the size of the flow going through a node. In other words, we want to ensure that nodes with a high volume of flow belong to large cuts. In other words, rather than seeking a uniform level of connectivity, we seek instead a uniform level of

### B. *Elasticity*

Regardless of how efficient is the original organization of the network, such organization is only statistically quasi-optimal. If there are no malicious attacks and nodes only fail because they exhaust their power it is possible that the network will remain connected until it collectively dies out. In other words, the network will deplete uniformly and die graciously chipping one bit at a time. Such scenario, while possible, is not very likely. There are enough unknowns to almost guarantee that whatever organization is selected in the beginning, it does not match exactly the scenario that will take place. For this, we want the organization to be dynamic and adaptive by allowing nodes from robust areas to cover for areas that get depleted. This is the concept of elasticity: whenever an area of the network is under excessive pressure, whenever possible, it should stretch rather break. In fact, elasticity consists of stretching much earlier than the time when the network is at the point of breaking. For example, consider the network in Figure 5 below. The flow goes from x to y, then z. Initially x, y, and z have cuts of size 3, 4, and 5 respectively. As 2 nodes from

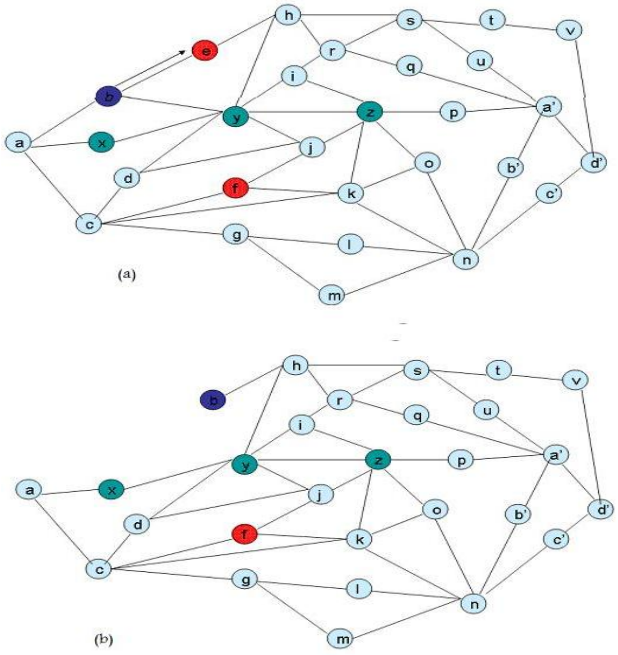



Figure 5. (a) nodes e and f fail they are part of cut set of node y. (b) node b moves to be part of node y's cut set

the cut of y fail, one node from the cut of x moves up and one node from the cut of z moves down resulting in cut sizes of 2, 4, and 5 (the nodes that migrates from z down to y is part of the cut of both y and z).

### C. *Differential Elasticity*

The concept of elasticity alone refers to the fact that when some pressure is exercised pulling on a surface from opposite directions, the components of the surface are rearranged as a response in order to avoid breaking the bonds between them and tearing apart. Some surfaces are elastic in all directions; others have a preferred (or exclusive) direction of elasticity. In this case, we want the elasticity to be in the direction of the flow. In other words the "stretching" of the network must be done primarily in parallel with the direction of the flow rather than transversally to it. This direction of movement of the nodes will be in line with where the highest need is likely to be and will also minimize un-necessary back and forth of the nodes as the network thins out.

## IV. ADAPTIVE ORGANIZATION

### A. *Autonomous Organization*

We recall that the objective is to have the nodes self organize so as to achieve a differential connectivity by giving a larger cut size to nodes with higher flow volume. We recall also that the intention is to use k*N nodes where only N would be sufficient for coverage and connectivity. We propose here a two-step process:

1. A sufficient number of nodes (e.g. 1.5*N) are deployed by spreading them using a uniform random distribution to ensure full coverage and connectivity.
2. This initial set of nodes collaborates to establish a reasonable flow pattern and identify the amount of flow traversing each of the nodes.
3. The remaining nodes are deployed in such a way that areas with high flow will attract a relatively large number of nodes, resulting in larger cuts whereas areas of small flow would attract fewer nodes resulting in smaller cuts.

We discuss each of these three stages:

1. Initial Uniform node deployment. This is the usual method to deploy most large sensor networks. The nodes are sprayed from a distance. A sufficient number of nodes are sprayed to maximize the likelihood that the network will cover the area of interest and will be connected. Because of the vulnerability of nodes (limited-life time battery, among others), most networks are deployed with sufficient redundancy. In other words, when only N nodes are sufficient to cover the area when placed in an optimal configuration, 2 (or more)*N nodes are placed. In this case, we target to provide enough redundancy to ensure connectivity and coverage, but we start with a number close enough to the minimal.

2. Establishing the flow pattern. A number of algorithms and approaches have been proposed to guide the pattern of communication (routing) between the nodes of a network. It is without loss of generality that we will assume that: 1. The

overall routing takes place using a routing tree, sub-graph of the overall network, 2. The tree is rooted at the base station or some other representative sink node, 3. The communication can be decomposed into two flows: from the root of the tree towards the leaf nodes to broadcast the "query", i.e., the nature of the data requested from the nodes and from the leaf nodes up towards the root to send the data back to the base station. 4. The two flows follow the same links in opposite directions. 5. Finally, we assume that a quasi-minimum spanning tree rooted at the sink node minimizing the total communication cost is a good approximation of the actual routing structure that will be used. The spanning tree will be constructed; the flow traversing each of the nodes based on this minimum spanning tree will then be used to estimate the actual flow going through those same nodes in the sensor network. The spanning tree has the advantage of being easily computed using a greedy distributed algorithm.

3. Deploying the remaining nodes. The key now is to deploy the remaining nodes so that they assemble around existing nodes in a way proportional to the flow through them. The general idea is to disperse the nodes and then subject them to attractive forces by the existing nodes whereby the intensity of the attractive forces is correlated with the flow in the nodes. One way to visualize this process is to see the nodes with their flow as points in 3D space where their positions on the plane represent their x and y coordinates and their flow represent their elevation (the elevation is in fact inversely proportional to the flow). Figure 6 shows the topography generated by a network flowing towards the bottom of the area.

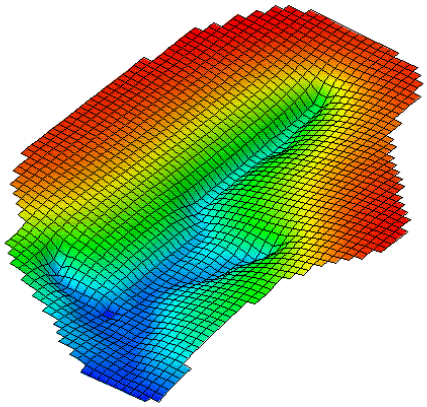


Fig. 6. Topography created by the network

In other words, whereas the first batch of nodes establishes the topography, the remaining nodes are dispersed and let initial random positions, gravity forces, and inertia determine their final positions. We discuss each of these three elements in turn:

Initial positions: In the same way that the initial batch was deployed at random to cover the area, we can do the same thing with the remaining nodes.

Gravity: The forces used to organize the node are calculated based on local information about gravity forces. The mobile nodes in this new batch have a node id, an x- and y- position, but have no flow (yet), i.e., they are not aware of their elevation. This latter information is obtained from the nodes in their communication range. Each node from the first batch broadcasts a message m=<topic= "position", id=self.id, x=self.x, y=self.y, z=self.f>. Each node from the new batch listens to all of the incoming messages, identifies nodes with the "lowest elevation", and moves towards them. The relative positions of the neighbors and their elevations determine the direction of movement and its intensity. More specifically:

- If there is only one neighbor, move towards that neighbor (this is unlikely to happen given the level of connectivity) only if its elevation is lower than self.
- If there are multiple neighbors with lower elevation than self, identify two neighbors with the lowest elevation, move towards the middle between them, calculate own elevation to be higher than the highest of the two.
- If there are multiple neighbors but all of them have a higher elevation then self, do not move.

Each of the movements prescribed above is a small step after which the node listens again and may keep moving.

Inertia: Using the gravity alone, nodes stop only when they reach local minima. Because the flow grows rather uniformly from the boundaries of the area towards the sink node, there is a risk that all the nodes will end up flocking towards the lowest elevation point, i.e., the sink node. We add an inertia force to ensure a more distributed spreading of the nodes. We use a repulsion force based on the crowdedness of the neighborhood. A node attracts mobile nodes when it has a low elevation. Also, a node also repulses mobile nodes with intensity proportional to the crowdedness of its neighborhood.

### *B. Adaptive Reorganization*

Assuming that we start with a reasonably good organization of the network, this organization may no longer be good after repeated failures of nodes. For sensor networks, node failures can be completely random with no correlation between the failure of a node and its position or the failure of its neighbors, or they may be relatively predictable (power depletion of a node with high level of activity), or correlated with the failure of other nodes (an animal tramping on a node may tramp on a cluster of nodes in the same neighborhood, or a malicious attacker who targets specific regions for the specific purpose of disrupting the network). Different patterns of failures require different approaches. Our proposal consists of three complementary approaches.

Differential connectivity initial organization. The whole idea of crowding the areas of intensive activity prolongs the life of the network by delaying its natural death (disruption)

 ISBN: 978-1-61208-109-0 

by exhaustion. This also protects areas that are more vulnerable to targeted attacks.

Differential Elasticity. Irrespective of how good the initial organization, eventually some regions of the network will weaken and eventually "break". Differential elasticity is the process by which mobile nodes keep a watch and move to where they are needed. Specifically, nodes are aware of the cuts to which they belong, and strategically pull other nodes towards them as their cut size is reduced. For example, consider Figure 5 again. Initially node x has a cut size 3 with some flow $f_x$ and node y has a cut of size 3 with a flow $f_y$. With the assumption that the nodes were distributed to ensure differential connectivity, the flow $f_x$ and $f_y$ must be comparable. When node y loses two nodes, e and f, from its cut, this generates an imbalance. Node y reacts by sending a jolt pulling nodes towards it to reinforce its cut. The force generated is characterized by the following features:

- Its direction is vector (x,y).
- Its intensity is proportional to the expression
- (size of cut (x))/(size of cut(y)) * $f_y/f_x$ -1

When the nodes are balanced, the force is zero. In the case of Figure 5, assuming $f_y/f_x$ =1, the force will have an intensity 3/1-1=2.

Notice that this force will be broadcast to the whole neighborhood and is likely to move multiple nodes closer to y. Some of these nodes may become part of the cuts of both x and y. The parameters dictating the exact intensity and duration of these jolts are determined experimentally so that they generate the right level of "elasticity". We want the structure to be sufficiently elastic to allow the necessary restructuring without being too over-reactive.

Global Restructuring through simulated annealing. In addition to the small local reorganizations, a less frequent more global re-organization is used to allow for better redistribution. The global reorganization is a form of simulated annealing whereby

1. Every mobile node is moved at random within a given distance (this is the heating part of simulated annealing).
2. A few iterations of the organization algorithm are then run to let the nodes cool down and settle back.

### C. Status and Future Work

While all the components of our proposed approach are in place, we are still in the process of validating it through simulation. In this simulation we set the length of the area, the communication and sensing range, we calculate the minimal number M of nodes needed for connectivity and coverage, and the multiple k that we would like to use. Given N=k*M, we start by placing a number slightly larger than M using a random placement. We then run the distributed greedy spanning algorithm to determine the flow from the nodes towards the sink node. The rest of the nodes are then placed randomly and subjected to the gravity and inertia forces. We have completed this stage of the simulation after iterative refinements and balancing between the gravity and the inertia forces. We now obtain a node distribution with a relatively uniform ratio cut over flow. Our current efforts are focused on the reorganization step especially that we are considering different types of scenarios for which we want the network to be resilient. These scenarios are (a) Scenarios of power consumption. One certain source of node failure is the depletion of its power. For this, we are experimenting with different types of queries: Queries that involve all regions uniformly; queries that are highly targeted requesting data from the same nodes repeatedly, and mixes of queries of the two types. We found that one of the characteristics of power depletion is that it happens gracefully—irrespective of the scenario. This is because power depletion is a continuous phenomenon that gives nodes time to the affected nodes to notify its neighbors that they may need to take action. (b) Scenarios of hardware failure. We are simulating hardware failure by picking nodes at random and declaring them faulty. As long as the death of the faulty nodes happens in areas that are sufficiently crowded, the network reacts as expected and makes up for the dead node. (c) Scenarios of non malicious accidents. We distinguish this scenario from the previous one by the fact that it is more localized in time and space. We are still experimenting with this scenario especially that we want to validate it with realistic data. Initial results we have obtained from the initial experimentations seem to confirm the well-foundedness of our approach. In particular, the networks organize in the way that we expect them to. For the scenarios that we have completed, the networks also reorganize as intended.

Additional refinements and work under way include:

(1) Refinement of the parameters used for the elasticity force and for the simulated annealing so as to optimize the triggers used for reorganization. We want to make sure that the network reorganizes soon enough to avoid a loss of functionality, but also that it does not re-organize prematurely and then undo work done later.

(2) Assessment of the extent to which the initial organization and the reorganization bring about any gains as compared to a fully random organization or some other organization leading to a uniform k-connectivity.

(3) Assessment of the reorganization. In particular, we will compute the distribution of the ratio size(cut)/flow for every node over time. Ideally we want to see that the variance remains relatively stable over time.

## V. CONCLUSION

Redundancy has always been the key ingredient used to provide fault tolerance. Sensor networks are no exception. Because in most applications sensor nodes have limited power and are hard to access, they are understood to be relatively vulnerable. The strength of sensor networks comes not from any single node, but from their number and from the way in which they communicate and collaborate.

 ISBN: 978-1-61208-109-0 

Redundancy is an integral part of sensor networks. The general idea being that if we put enough nodes, the network should be resilient since whatever nodes die there will be others in their neighborhood. The concepts of k-connectivity have been used in the recent years to quantify this redundancy and fault tolerance. The concept of k-connectivity for example states that in a k-connected network, the network will remain connected after the failure of *any* k-1 nodes. While very useful, this concept does not fully capture how fault tolerant a network is. In particular, we want to know "how likely is a k-connected network to be disconnected after the failure of k, or k+1, or 2k nodes?" Another important and more pragmatic question is the one we address here: Suppose that I am willing to devote h times more nodes than needed. What is the best use of these additional nodes? Should we organize them so that the network has the highest k-connectivity level? We argue here that instead of uniform connectivity, we would be better off increasing the connectivity where it matters the most rather than uniformly. Furthermore, we propose distributed algorithms that allow the nodes to organize themselves in an autonomous manner and to reorganize themselves as dictated by the changing needs of the network. The proposed algorithms are based on forces exercised by nodes over other nodes in their communication range. The interplay between the different forces generates the desired collective behavior. Initial results from the simulation validate our approach in the sense that we are able to generate the desired behavior repeatedly under different scenarios. We are in the process of develop metrics that can capture the behavior in a more objective manner and generating scenarios that would allow us to compute the gain in energy and lifetime as compared with other approaches that use different initial organizations and criteria and different (or no) re-organizations.

## REFERENCES


[1] Al-Karaki, J.N., Kamal, A. E. 2004. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications*, Dec. 2004, Vol 11, No. 6, pp.6-28, Dec. 2004.

[2] Alzoubi, K. M., Wan, P., and Frieder, O. 2002. Message-optimal connected dominating sets in mobile ad hoc networks. In*Proceedings of the 3rd ACM international Symposium on Mobile Ad Hoc Networking &Amp; Computing* (Lausanne, Switzerland, June 09 - 11, 2002). MobiHoc '02. ACM, New York, NY, pp. 157-164. DOI= http://doi.acm.org/10.1145/513800.513820.

[3] Ammari, H. M. and Das, S. K. 2009. Fault tolerance measures for large-scale wireless sensor networks. *ACM Transaction on Autonnmous and Adaptive Systems* 4, 1 (Jan. 2009), pp. 1-28 doi= http://doi.acm.org/10.1145/1462187.1462189.

[4] Bai, X., Kumar, S., Xuan, D., Yun, Z., and Lai, T. H. 2006. Deploying wireless sensors to achieve both coverage and connectivity. In *Proceedings of the 7th ACM international Symposium on Mobile Ad Hoc Networking and Computing*(Florence, Italy, May 22 - 25, 2006). MobiHoc '06. ACM, New York, NY, 131-142. DOI= http://doi.acm.org/10.1145/1132905.1132921.

[5] Bredin, J. L., Demaine, E. D., Hajiaghayi, M., and Rus, D. 2005. Deploying sensor networks with guaranteed capacity and fault tolerance. In *Proceedings of the 6th ACM international Symposium on Mobile Ad Hoc Networking and Computing* (Urbana-Champaign, IL, USA, May 25 - 27, 2005). MobiHoc '05. ACM, New York, NY, pp. 309-319. doi= http://doi.acm.org/10.1145/1062689.1062729.

[6] DelaRose, J. Y. Liu, L. Mili, A. Phadkee, L. Davilva, 2005. Catastrophic failures in power systems: Causes, Analyses, and Countermeasures. Invited paper. *Proceedings of the IEEE*. Vol. 93, No. 5, pp. 956-964.

[7] Jia, X, Kim, D., Makki, S. Wan, P-J, and Yi, Ch-W. 2009. Power Assignment for k-connectivity in Wireless Ad Hoc Networks. *Journal of Combinatorial Optimization*, Springer, Netherlands, Vol 9, No 2, pp. 213-222. Doi=http://dx.doi.org/10.1007/s10878-005-6858-2.

[8] Khelifa, B. Haffaf, H. Madjid, M. and D. Llewellyn-Jones 2009. Monitoring Connectivity in Wireless Sensor Networks. *International Journal of Future Generation Communication and Networking* Vol. 2, No. 2, June, 2009

[9] Lambrou, Th., Panayiotou, Ch., Felici, S., Beferull, B. 2010 Exploiting Mobility for Efficient Coverage in Sparse Wireless Sensor Networks. Wireless Personal Communications, Vol 54, pp. 187-201, doi:10.1007/s11277-009-9717-0.

[10] Mili, A. Self-Stabilizing Programs: The Fault-Tolerant Capability of Self-Checking Programs, IEEE Transactions on Computers, pp. 685-689, July, 1982.

[11] Schmid, S. and Wattenhofer, R. 2006. Algorithmic models for sensor networks, *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International* , pp. 25-29 April 2006 doi: 10.1109/IPDPS.2006.1639417.

[12] Wu, Y. and Li, Y. 2008. Construction algorithms for k-connected m-dominating sets in wireless sensor networks. In *Proceedings of the 9th ACM international Symposium on Mobile Ad Hoc Networking and Computing* (Hong Kong, Hong Kong, China, May 26 - 30, 2008). MobiHoc '08. ACM, New York, NY, pp. 83-90. doi= http://doi.acm.org/10.1145/1374618.1374631.

[13] Xing, G., Wang, X., Zhang, Y., Lu, Ch., Pless, R., and Ch. Gill. 2005. Integrated coverage and connectivity configuration for energy conservation in sensor networks, *ACM Transactions on Sensor Networks (TOSN)*, v.1 n.1, p.36-72, August 2005.

[14] Zhang, H. and Hou, J. 2004. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks. In NSF International Workshop on *Theoretical and Algorithmic Aspects of Sensor, Ad Hoc Wirelsss, and Peer-to-Peer Networks*, 2004.


 ISBN: 978-1-61208-109-0 

# Implementation Architectures for Adaptive Workflow Management

Hanna Eberle, Frank Leymann and Tobias Unger
*Institute of Architecture of Application Systems (IAAS)*
*University of Stuttgart*
*Stuttgart, Germany*
*email: eberle@iaas.uni-stuttgart.de, leymann@iaas.uni-stuttgart.de, unger@iaas.uni-stuttgart.de*

***Abstract*—Business processes are often required to be highly flexible and adaptive due to the fact, that business conditions change. Therefore, there exist a lot adaptation and flexibility concepts for workflows. However, workflow adaptation concepts are often discussed on the language level neglecting a discussion on the implementation architectures. Until now, effective implementation architectures have not been investigated. Therefore, the main contribution of this work is to research three implementation strategies for adaptive workflow management, which we discuss with respect to modeling requirements and change management.**

***Keywords*-Adaptable Workflows, Flexible Workflows, Workflow Execution Architecture**

## I. Introduction

Adaptation concepts in the workflow technology are manifold. Since business conditions change, business processes are often required to be highly flexible and adaptive. Workflow execution flexibility is a key enabler for workflow adaptation, because if a system does not provide any means for flexibility the system cannot adapt. In the workflow domain, the terms workflow flexibility and workflow adaptation are often used synonymously. In this work we also discuss workflow adaptation from the flexibility point of view. Furthermore, we distinguish between workflow adaptation and workflow evolution. The changes of the business logic become visible to one instance only in workflow adaptation, while the change of a workflow model in terms of workflow evolution affects all running instances. Traditional workflow languages provide means to specify partial orderings between units-of-work, which are also called activities. Most workflow languages rely on graph theory [12] and define the partial orderings between the activities as directed graphs, where the edges in these workflow graphs are called control connectors. Today, applications are often realized as Web Services following the SOA paradigm and workflows orchestrate the different Web Service applications to become a new and more complex application. Therefore, activities send and receive messages to interact with the activity implementations represented as Web Service [18]. Execution flexibility in workflows can be established in two ways, either the partial ordering of activities or the set of activities is adapted, which is also called adaptation of business logic, or the selection and binding of activity implementations is performed at runtime and therefore provides a point of flexibility. In this paper we focus on the changes of business logic.

The state-of-the-art in the domain of business logic adaptation of workflows, however, focuses on the change and change management of workflow models on a language and conceptional level, defining business logic change operations and investigation on correctness criteria for the correct changes for a certain set of instances [14] [2].

Until now, effective implementation architectures have not been investigated. Most adaptation approaches such as process fragments [5] are implemented using instance migration c.f., [4]. However, instance migration is not a sufficient implementation approach for the process fragment-based adaptation approach as it requires e.g. to suspend the process execution while the migration. Other adaptation implementations for adaptive workflows are feasible, which are implementing the process-fragment adaptation concept more natively and overcome some disadvantages of the instance migration implementation approach. Therefore, the aim of this work is to research and discuss new implementation architectures for adaptive workflow management.

We present three adaptation implementation architectures for adaptive workflow management of internal workflow artifact adaptation and discuss these approaches with respect to modeling requirements and change management.

The paper is organized as follows. First, in Section IV, we discuss the related work runtime adaptation concepts in the workflow domain for graph-based workflows. We examine the workflow adaptation concepts from a modeling language perspective and the implementation concepts subsequently, that are available to implement the workflow adaptation concepts. Secondly, we present and discuss adaptation implementation architectures, which realize one adaptation implementation concept in Section V. The adaptation implementation architectures handle adaptation internally by adapting the internal model artifacts. We conclude our work in Section VI.

## II. Traditional Workflow Execution

### A. Execution Architecture

Workflow languages are programming languages that are interpreted, like e.g., Java [6]. Source code, written in interpreter programming languages, are not complied into machine

 ISBN: 978-1-61208-109-0 

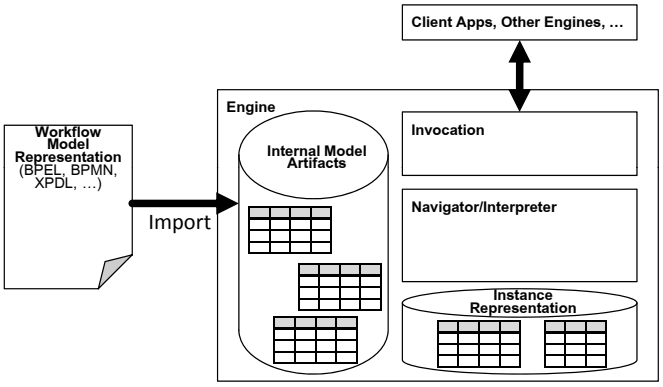



Figure 1: WFMS Architecture (c.f. [7])

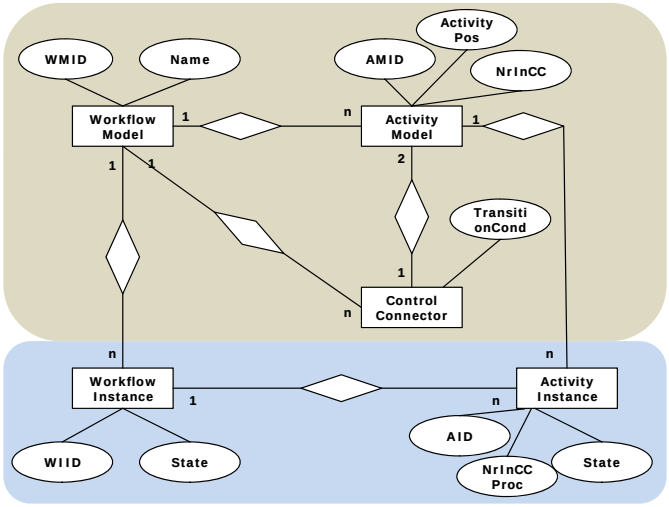



Figure 2: ER-Diagram (c.f. [12])

code, which can be directly run by a machine without any other additional components or programs. The source code must be complied into the platform specific source code. Once it is complied it cannot be run on another platform, but must be complied using another platform specific complier. Source code of interpreter languages are translated into a representation, e.g., byte-code, which is run by a interpreter or a virtual-machine, where the interpreter is platform specific, but the source code representation is platform independent. Workflow models act like source code, which get imported and deployed to a virtual-machine for workflow execution, which is widely called Workflow Management System (WfMS). A simplified WfMS architecture based on [7] [12] is presented in Figure 1. Workflow models are represented in a workflow language, like the XML based language BPEL [13]. At deployment time the workflow models are imported and translated into internal artifacts [11] (c.f. 2), e.g., table entries of a relational representation [12] or Java objects. The internal artifacts are interpreted by the Navigator Component. In particular, workflow languages are designed to prescribe orderings between activities. Activities represent business activities, basically, where the activity implementation of an activity might be a client application implemented in another programming language or even implemented by another process. The sending and receiving of messages is handled by the Invocation Component. Workflow models are imported, activated and deactivated using workflow model management API functions. The workflow instance API functions manage the state of a workflow instance, e.g., to suspend and resume a workflow instance as well as a function to terminate a workflow instance.

### B. Internal Representation

A simplified schema for the relational representation of workflow models is presented in Figure 2. A workflow model consists of a set of activity models, which are connected using control connectors. An activity model is aware, whether it is a start or an end activity of the workflow, which is stored in the ActivityPos attribute. The NrInCC attribute represents the amount of incoming control connectors. A control connector can be annotated by a transition condition, which is a function mapping to a Boolean value. Each process model might have more than one concurrently running process instance. Workflow instances are represented by the state of the workflow instance and the states of its activity instances. The activity instance attribute NrInCCProc keeps track with the amount of already evaluated incoming control connectors.

### C. Navigation

A workflow instance is created applying one of the instantiation patterns as presented in [3], e.g., a message or an event is received at an endpoint of a start activity. The Navigator uses the relational representation of the workflow model and the state of the workflow instance to determine the next activity to be executed. An activity is executed, if all incoming control connectors are evaluated. The activity instance keeps track with the amount of evaluated control connectors by increasing the amount of the attribute NrInCCProc by one each time the activity is target of an evaluated control connector. After an activity is finished the associated outgoing control connectors are selected, evaluated. A workflow instance completes successfully, if all end activities are finished.

## III. Adaptation Example

To provide a more concrete idea of the business logic adaptation, we present a little example on what shall be achieved by business logic adaptation. The adaptation of the business logic in workflow models influence the produced execution traces of the workflow instances. An execution trace is denoted by the execution ordering between the activity instances. Our example workflow model consists

 ISBN: 978-1-61208-109-0 

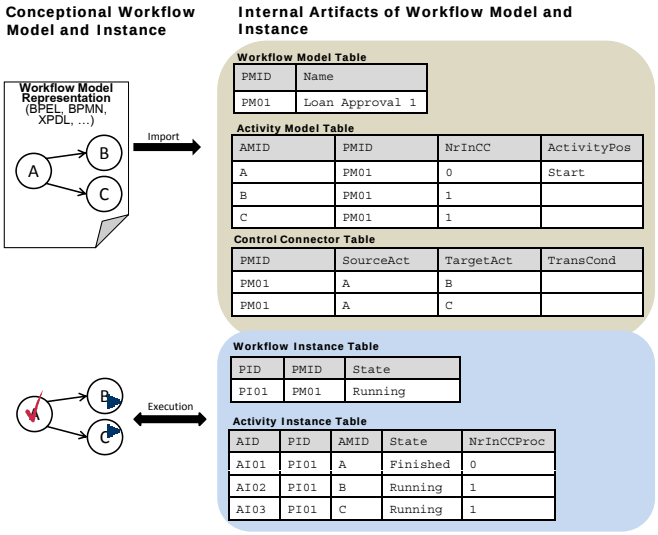



Figure 3: An Example

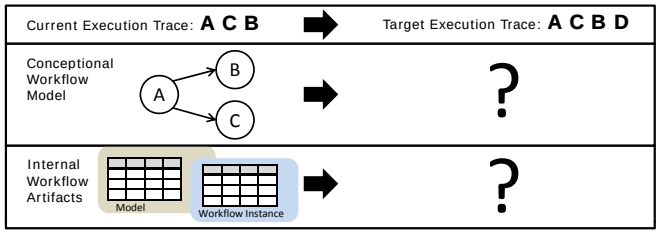



Figure 4: The Problem

of three activities A, B and C. A workflow instance of the workflow model has two running activities, B and C. A possible conceptional and internal representation of the example is depicted in Figure 3. The execution trace of that instance is *ACB*. The target execution trace of the adaptation is *ACBD*. An activity D shall be executed additionally, with the constraint that the activity D must be executed after the completed execution of activity C. Two questions arise from this little example of workflow adaptation. How is the workflow model modeled and adapted conceptionally, and how is the workflow model adapted internally? These questions are discussed in the remainder of this paper.

## IV. Related Work

Adaptation approaches are manifold in the workflow domain. To be able to classify the approaches we investigate the approaches by regarding two aspects: (i) How is the adaptation achieved from a modeling point of view? (ii) What are the possible implementation strategies for these adaptation concepts?

### A. Modeling Adaptive Workflows

There exist two basic adaptation approaches for graph-based workflow modeling languages. These approaches denote the ends of a spectrum. Hybrid approaches are possible. The first adaptation approach acts on the assumption, that a complete process model is modeled at design time. At runtime the process model needs to be adapted, due to the fact, that e.g., the business conditions might have changed. Activities are added or removed by the adaptation or the activity ordering is changed. The changes in the model are either applied to all instances of the process model, or made visible to one instance only. Approaches, that pursue that modeling approach, are presented in [2] [15]. The second adaptation approach acts on the assumption, that parts of a comprehensive process model are known at modeling time, e.g., by different parties. These parties are able to model the parts of a comprehensive process model. The comprehensive process model is assembled of these process parts. The assembling may currently available knowledge in terms of context data into account. That way the comprehensive process model is created adaptively, as the selection of the to be integrated parts and the integration of the different parts can be based on adaptation relevant data. The comprehensive process model creation might be performed either at runtime or at design time. If the selection and integration is performed at runtime, even runtime data can be employed by the selection. There exist various modeling approaches for the process part modelling, e.g., such as subprocesses [10], worklets [17], proclets [1] [16], process fragments [9].

### B. Adaptation Implementation Strategies

In this section, we discuss the possible implementation strategies for the adaptation concepts. Afterwards, we investigate, which conceptional adaptation concept can be implemented by which of the implementation concepts, since not every adaptation concept can be implemented employing every implementation concept. We could identify two different adaptation management strategies for adaptation of graph-based workflows. The first implementation approach is to manage adaptation of the conceptional workflow model as a change of the internal model. The change of the workflow model is mapped to the internal artifacts as well. This strategy can be applied to both adaptation modeling approaches presented previously. This approach needs to be aware, whether the change shall be applied to one instance only or to all instances running on that model. In the case a model is adapted the change must comply to the execution traces of the instances, which are about to be adapted and run on the new conceptional model. Both adaptation modeling approaches can be implemented using this implementation strategy. Implementation architectures for this adaptation implementation strategy are discussed in subsequent sections of this paper. The second adaptation implementation strategy does not change the internal representation of the model at all. The adaptation of the workflow model, e.g., the composition of a part of a process with another part of a process is

 ISBN: 978-1-61208-109-0 

| | Conceptional change of Model | AOP | Process Composition |
|---|---|---|---|
| Change of Internal Model | ✓ | ✓ | ✓ |
| No Change of Internal Model Map to Interactions | | ✓ | (✓) |

Figure 5: Overview

mapped to interactions between the processes, like e.g., subprocesses, where the subprocess is integrated into the parent process by sending messages to the subprocess, which might be running in a different process context. However, the interaction must be implemented by a complex coordination protocol to be able to make the interacting processes appear to the outside as one process. The interaction also requires the interaction points to be modeled. This approach is not applicable, if the changes of the conceptional model are not modeled as a separate model or if parts of a process part need to be adapted and the interaction points are already modeled in the to be integrated parts. Changes in the pre-modeled parts of a process model can only be implemented by an implementation architecture, which is able to change the corresponding internal artifacts as well. Whereas the composition of pre-modeled parts can sometimes be mapped to interaction relations or always by implemented by changing the internal artifacts of the internal model adding new control connectors to glue the pre-modeled parts together. Whether the interaction approach can be applied, depends on the pre-modeled process parts and the integration points and operations, that are provided by the process part. The integration approach in not that flexible as the internal change or model approach, since the points of integration must be pre-modeled. The summary of the discussion is presented in Figure 5.

## V. Adaptation Architectures applying the Internal Changes Strategy

In the following, we introduce adaptation architectures, where the conceptional changes of the workflow model are mapped onto changes of the internal workflow representation. We do not answer the question, whether an adaptation step is correct for an instance, which is discussed in various papers, e.g., in [2], [14]. We discuss implementation architectures for adaptation. The first implementation architecture supports the instance migration concept as presented in WebSphere MQ Workflow [8]. Instance migration associates an instance of a workflow model with another workflow model. The second implementation architecture supports the adaptation of a process model natively. The third and last approach presented in this paper supports the composition of pre-modeled process parts natively. We base our discussion on the internal workflow model and instance representations as presented in Section II.

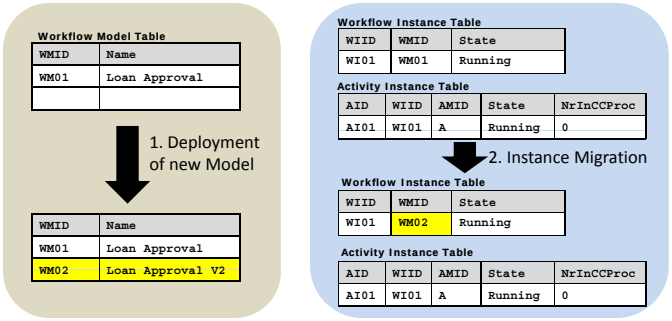



Figure 6: Instance Migration

### A. Adaptation Architecture supporting Instance Migration

Instance migration associates a suspended workflow instance with another workflow model, where the target workflow model must fit to the already executed parts of the instance, and resumes the execution. Adaptation approaches applying instance migration are discussed e.g., in [2]. The concept of instance migration can be applied to implement adaptation, thus adaptation can be mapped instance migration, because the old workflow model can be changed and imported as new workflow model, which becomes the new workflow model of the instance. The internal artifacts are influenced by the migration as depicted in Figure 6. The workflow instance identifier, WIID, of the instance keeps the same, but the workflow model identifier, WMID, is changed to the WMID of the new workflow model. The traditional APIs can be used to import the new workflow model to the WfMS as well as to suspend and resume the workflow instance. A new migration API function needs to be added to be able to select and migrate a specific instance to a new workflow model. The instance migration approach allows other workflow instances associated with the old model. Since only the selected instance is migrated and the old workflow model is not changed, the remaining instances can finish on the old workflow model. This approach is suitable, if the adaptation of a workflow instance is an exceptional case. The adaptation is percepted to the outside, since the workflow instance is now associated with another workflow model. However, message correlation issues must not be handled differently, since the correlation is based on business case data, which is associated with the instance.

The perception of the process now might be different from the one perception before the adaption step, since it might be possible that new activities are deployed, which receive and send messages.

### B. Adaptation Architecture supporting Changes of the Internal Representation

Another implementation approach for adaptable workflows is to change the workflow model directly, both the workflow model and the internal representation as presented in Figure

 ISBN: 978-1-61208-109-0 

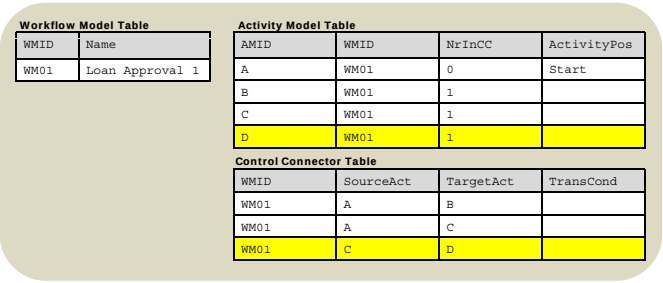



Figure 7: Extending the Model Example

7. The workflow model is extended by the adaptation logic. The new activities are deployed and the new activities and control connectors are written into the workflow model tables. This approach has the advantage, that only the workflow model is changed as depicted in Figure 7, while nothing needs to be changed at the instance level. This approach is only applicable, if all instances running on the changing model comply to the change. This approach is suitable, if a workflow model needs to be changed very often and the workflow model has only very few instances running on it. This approach suits best for late modeling approaches and ad hoc workflows, if the relationship between model and instance is one-to-one, where the repetition rate is very low and the workflow model must be highly flexible. The same new API function needs to be added to the API function set as for the instance migration. Input to this function is the workflow model ID of the workflow model to be changed and the new workflow model. Internally, the function defines the new parts of the workflow model and deploys these parts and maps them to the internal representation to extend the workflow model tables.

### *C. Adaptation Architecture using Inter-Workflow-Model Control Connectors*

If the target workflow model of the adaptation is a composition of already deployed workflow parts, which are already represented internally, we are able to map the target conceptional workflow model internally by introducing ad hoc control connectors. These control connectors connect activities of two different workflow models. However, this control connector is visible to one instance only, since the composition is only valid for one instance. The ad hoc control connector is hybrid, since it represents model information, but is only evaluated by one instance. This idea of this approach is quite close to the implementation idea of the interacting implementation strategies. However, no complicated coordination protocols must be executed to coordinate the executions of the different parts of the overall process, because the instance is not distributed over many execution contexts. The whole instance is executed within the same execution context and the complete workflow model is represented internally by the insertion of ad hoc

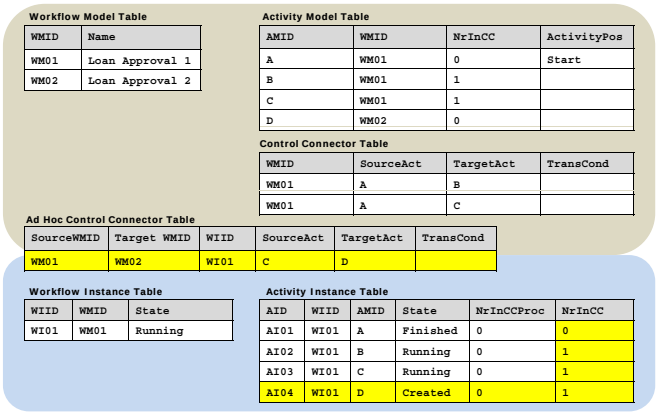



Figure 8: Connecting two Models by the Definition of an Ad Hoc Control Connector

control connectors as presented in Figure 8. The insertion of the ad hoc control connector changes the amount of incoming control connectors of the target activity. Therefore, we extend the activity instance table by a NrInCC column, which represents the amount of common incoming control connectors plus the amount of incoming ad hoc control connectors. The activity instances of a workflow part are created after this workflow part has been integrated into the workflow and the NrInCC is set during creation. A standard navigator must be adopted to be able to navigate over these workflow models and ad hoc control connectors. Therefore we first navigate the standard control connectors of the source activity afterwards we navigate over the ad hoc control connectors. Since all process parts are pre-modeled, the pre-modeled parts do not change and therefore the perception of the parts and interaction possibilities do not change. The most extreme case in this realization is to model all control connectors as inter process control connectors and each activity itself is a process model. Also this approach has its restrictions, since e.g., activities cannot be removed.

### *D. Discussion*

The adaptation implementation architectures are powerful adaptation implementation strategies. These adaptation architectures enable to adapt a executed workflow model either by adding or removing parts of the workflow model. However, each of the implementation strategies is optimized towards a specific modeling paradigm and application scenario. The instance migration strategy was introduced first to manage versioning issues. Instance migration requires a smart model management due to the fact, that the amount of very similar workflow models grows. Instance migration requires the process instances to be suspended, before the instances can be migrated to the new version of the process model. The instance migration implementation concept is especially suitable to handle exceptional cases. The happy path is modeled in the original process model and in case a fault

 ISBN: 978-1-61208-109-0 

occurs the faulting instance is drawn onto another process model, which is able to deal with the faulting situation. Faults are usually unexpected and therefore its not possible to cover all these faults by a process model. Whereas the native adaptation support implementation architecture should be applied for workflows, which are known to be adapted in an ad-hoc and therefore only one instance is created per natively adapting workflow model. The instance does not need to be suspended to perform the adaptation of the workflow model. Both of these approaches, the instance migration and the native ad-hoc changes of the model, change the set deployed activities and therefore might also change the set of activity message endpoints. Therefore the perception of the workflows are changed, since they now provide more endpoints, that can be called. The third implementation architecture is optimized and tailored towards adaptation concepts similar to the process fragment concept. The local knowledge is modeled locally and deployed to an engine. Depending on the runtime context data the parts are glued together by introducing instance control connectors. The process instance needs not to be suspended and no new activities need to be deployed. The perception of the process fragments does not change to the outside.

## VI. Conclusion

In this work, we analyzed modeling and implementation approaches, which realize adaptation of business logic of workflows. We presented three implementation architectures for adaptive workflow management. The adaptation is handled by changing the executed model internally. We discussed the applicability of each of these implementation architectures and we argued, that the applicability and suitability of the implementation strategy chosen depends on the scenario that shall be implemented. The scenario specifies e.g., what information is known at modeling time, or what information is pre-modeled. In our future work we will evaluate the concepts as presented in this paper implementing all the approaches and compare these approaches with the already implemented instance migration approach.

## Acknowledgment

This work is partially funded by the ALLOW project. ALLOW (`http://www.allow-project.eu/`) is part of the EU 7th Framework Programme (contract no. FP7-213339).

## References

[1] M. Adams, A. H. M. ter Hofstede, D. Edmond, and W. M. P. van der Aalst. Worklets: A Service-Oriented Implementation of Dynamic Flexibility in Workflows. In R. Meersman and Z. Tari, editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 291–308. Springer, 2006.

[2] F. Casati, S. Ilnicki, L.-j. Jin, V. Krishnamoorthy, and M.-C. Shan. Adaptive and Dynamic Service Composition in eFlow. In *CAiSE '00*, pages 13–31, London, UK, 2000. Springer.

[3] G. Decker and J. Mendling. Instantiation Semantics for Process Models. In *BPM '08*, pages 164–179, Berlin, Heidelberg, 2008. Springer.

[4] H. Eberle and S. Telezhnikov. *JBPM Fragment Engine*, 2010. http://code.google.com/p/jbpm-fragment-engine/.

[5] H. Eberle, T. Unger, and F. Leymann. Process Fragments. In *Proceedings CoopIS 2009*, pages 398–405, Berlin, Heidelberg, 2009. Springer.

[6] J. Gosling, B. Joy, and J. S. Guy L. *The Java(tm) Language Specification*. Addison-Wesley Longman, Amsterdam, June 2005.

[7] D. Hollingsworth. Workflow Management Coalition - The Workflow Reference Model. Technical report, Workflow Management Coalition, Jan. 1995.

[8] IBM Corporation. *IBM MQSeries Workflow: Concepts and Architecture*, 2006.

[9] K.-H. Kim, J.-K. Won, and C.-M. Kim. A Fragment-Driven Process Modeling Methodology. In O. Gervasi, M. L. Gavrilova, V. Kumar, A. Laganà, H. P. Lee, Y. Mun, D. Taniar, and C. J. K. Tan, editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 817–826. Springer, 2005.

[10] O. Kopp, H. Eberle, F. Leymann, and T. Unger. The Subprocess Spectrum. In *Proceedings of the Business Process and Services Computing Conference: BPSC 2010*, Lecture Notes in Informatics. Gesellschaft für Informatik e.V. (GI), September 2010.

[11] F. Leymann. BPEL vs. BPMN 2.0: Should You Care? In *2nd International Workshop on BPMN*, Lecture Notes in Business Information Processing. Springer, October 2010.

[12] F. Leymann and D. Roller. *Production Workflow - Concepts and Techniques*. PTR Prentice Hall, January 2000.

[13] OASIS. Web Services Business Process Execution Language Version 2.0. Committee specification, OASIS Web Services Business Process Execution Language (WSBPEL) TC, Jan. 2007.

[14] M. Reichert and P. Dadam. $ADEPT_{flex}$-Supporting Dynamic Changes of Workflows Without Losing Control. *J. Intell. Inf. Syst.*, 10(2):93–129, 1998.

[15] M. Reichert, S. Rinderle-Ma, and P. Dadam. Flexibility in process-aware information systems. *T. Petri Nets and Other Models of Concurrency*, 2:115–135, 2009.

[16] W. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer. Workflow Modeling using Proclets. In *CoopIS00*, pages 198–209. Springer, 2000.

[17] W. M. P. van der Aalst, P. Barthelmess, C. A. Ellis, and J. Wainer. Proclets: A Framework for Lightweight Interacting Workflow Processes. *Int. J. Cooperative Inf. Syst.*, 10(4):443–481, 2001.

[18] S. Weerawarana, F. Curbera, F. Leymann, T. Storey, and D. F. Ferguson. *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2005.

 ISBN: 978-1-61208-109-0 

# Interactive Access Rule Learning: Generating Adapted Access Rule Sets

Matthias Beckerle, Leonardo A. Martucci, Sebastian Ries
*Technische Universitt Darmstadt*
*Darmstadt, Germany*
{*beckerle, leonardo, ries*}*@tk.informatik.tu-darmstadt.de*

***Abstract*—This paper tackles the problem of usability and security in access control mechanisms. A theoretical solution for this problem is presented using the combination of automatic rule learning and user interaction. The result is the interactive rule learning approach. Interactive rule learning is designed to complete attribute-based access control to generate concise rule sets even by non-expert end-users. The resulting approach leads to adaptive access control rule sets that can be used for smart products.**

***Keywords-adaptivity*; *usability*; *access control*; *rule learning*.**

## I. INTRODUCTION

Smart products are a new class of devices that bridge the gap between the real and the virtual world. They provide a natural and purposeful product–to–human interaction and context-aware adaptivity. Smart products need to have knowledge about the application and environment that they are immersed to fulfill their tasks. Thus, they also need access to private / confidential information, such as users' preferences. Moreover, smart products can exchange private / confidential information among each other to complete collaborative tasks that require information from multiple sources, such as the booking of flight tickets and hotels. Smart products can be part of highly dynamic environments, where devices can appear and disappear in non-predictable ways.

However, the amount of possible security breaches is directly proportional to the sheer number and variety of smart products. Equally, the variety of devices with different user interfaces also increase the complexity of administrative tasks for the end-users. Therefore, one of the main challenges of IT-security regarding smart products is the design of mechanisms that combine a customizable level of security and usability [1], [2].

Current IT-security solutions tend to overstrain non-expert users. In home and enterprise environments, users are frequently forced to choose passwords for local and remote authentication and also define rules for access control, e.g., file sharing access rights. However, the imposition of such security features often lead to insecure or unpractical measures, such as written passwords and access control rules that are often too general. In addition, users tend to deactivate security mechanisms or render them useless by: not changing default passwords or leaving them blank; granting access to everyone; or turning off basic security mechanisms. This behavior is very common nowadays, especially regarding login passwords, browser cookies, virus scanners, and file access controls.

The administration of secure features in computational systems by non-expert end-users is already a challenge. Such a fact can be easily shown by the massive number of computers that are part of bot nets [3], which is, in most of cases, caused by inability of such users to keep their systems up-to-date or to change default settings. Smart products add more complexity to such scenarios by increasing the administrative burden to the end-users.

In this paper, the usability aspects of security solutions are analyzed and a mechanism that allows more user-friendly access control rules generation is proposed. The initial analysis is used to identify usability gaps in basic security mechanisms that can be applied for smart products. Such an analysis shows that there are already sufficient solutions that can applied for confidentiality, integrity, and authentication services, but no appropriate access control solutions exist nowadays.

This paper is organized as follows. In Section II, some key terms are defined. Section III brief outlines the state-of-the-art solutions for maintaining confidentiality, integrity, authentication, and authorization in highly dynamic environments. In particular, authorization and access control mechanisms are analyzed in Section IV, and security and usability requirements for access control rule set are introduced in Section V. In Section VI, we present a solution that helps users to generate proper access control rule sets using a combination of automated rule learning and user interaction. The related work is shown in Section VII. A use case scenario that illustrates our proposal is described in Section VIII, and the final remarks are presented in Section IX.

## II. DEFINITION OF SECURITY TERMS

In this section, we define the some key security terms that are going to be use throughout this paper: reliability, usability, confidentiality, integrity, authenticity and authorization.

- **Reliability**: in this paper we define reliable security as a set of security mechanisms that is able to fulfill the security expectations of an end-user regarding their security requirements.

 ISBN: 978-1-61208-109-0 

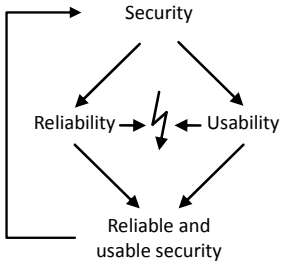



Figure 1. Dependencies for reliable and usable security

- **Usability**: in this paper usability means that security mechanisms demand minimum user interference to be deployed. A smart product should stay as usable as it would be without security mechanisms. Thus, the introduction of security should be preferably automated.
- **Confidentiality**: means that the assets of a computing system are accessible only by authorized parties. Confidentiality is usually implemented using cryptographic algorithms.
- **Integrity**: means that assets can be modified only by authorized parties or only in authorized ways. Integrity is mostly implemented using one-way functions in combination with cryptographic algorithms.
- **Authenticity**: means that an entity can prove who or what they claim to be. Authentication services are usually implemented by a proof of knowledge, a proof of ownership, or a proof of biometric trait.
- **Authorization**: means that policies are used and enforced to specify access rights. Authorization is implemented through access rules that are used by access control mechanisms to determine if an entity is allowed to access information or not.

The aforementioned terms reliability and usability are often seen as contradicting goals, especially regarding access control rules. Such contradiction is usually resulting from the huge amount of rules that are required to secure a system, which makes them unintelligible for end-users. Usability in most cases is simply neglected, what can result in insecure systems in the long-term since users tend to turn such security features off or use them in improper ways, as mentioned in Section I. These dependencies are shown in Figure 1. In the remainder of this paper the term CIA is used as an abbreviation for the security services confidentiality, integrity, and authentication[1].

## III. CIA AND SMART PRODUCTS

To achieve reliable and usable security, an analysis of existing security services in the context of smart products and highly dynamic environments is needed first. This section presents such an analysis. In such a context, we show that confidentiality, integrity and authenticity can be automated quite well, but authorization cannot. Confidentiality is presented in Section III-A, integrity in Section III-B, authenticity in Section III-C, and authorization in Section III-D.

### A. Confidentiality

For a reliable secure system it is important to secure not only the access to the data, but also to secure the data itself, whereas stored or in transit. Confidentiality can be achieved using encryption to protect data.

There are symmetric, such as AES, and asymmetric encryption mechanisms like RSA . Symmetric key encryption demand the distribution of cryptographic keys among participating devices. Asymmetric key encryption performs, in general, worse than symmetric key encryption. Hence, large chunks of data are rarely encrypted using asymmetric keys, but only selected data, such as symmetric keys. In smart products, the process of symmetric key distribution is a potential challenge because if a unique key is demanded for every pair of communicating entities, the number of required keys equals $\binom{n}{2}$, where $n$ is the total number of communicating devices. Nonetheless, it is feasible to embed public-private key pairs into them. Such an approach is sufficient in principle and implements confidentiality into high dynamic environments using existing and standard cryptographic systems.

### B. Integrity

Integrity has to assure that any unauthorized change of data is recognized. Data integrity is usually accomplished using one-way hash functions and public key encryption or with just symmetric keys. Message Authentication Codes (MAC) [4] are implemented using symmetric keys and digital signatures with public-private key pairs. Since such cryptographic tools are expected to be embedded into smart products (as seen in Section III-A), there are going to be enough cryptographic tools available for securing data integrity.

### C. Authenticity

Authentication is required to obtain a proof of correctness over an identity claim. In smart product scenarios there are basically three types of authentication: device–to–device, device–to–user, and user–to–user. There are sufficient mechanisms based on digital certificates that can carry out device–to–device authentication automatically. Device–to–user and user–to–user authentication can also be realized using proofs of knowledge, biometric traits or digital tokens together with public-key encryption. In such a case, after users authenticate themselves to smart products, such devices might be used to automatize other authentication procedures between users and other devices.

[1]In this paper CIA deviates from the more common usage of the term, which refers to confidentiality, integrity, and availability.

 ISBN: 978-1-61208-109-0 

### D. Authorization

Authorization is needed to specify access rights and enforce them. It is implemented through access rules, and the collection of such rules is referred to as a rule set. There are mechanisms that allows fully automated generation of rule sets for smart products. Such approaches, however, disregard adaptivity to the end-user. The general problem is resulting from the diversity of user preferences, so more information regarding the users is required. Authorization problems regarding adaptivity and user in smart products are discussed in Section IV, where the existing access control models are outlined and evaluated regarding their suitability to smart product scenarios.

## IV. Access Control

This section provides an overview of different access control models and provides an evaluation of such models regarding their suitability to smart product scenarios. In this section, we describe the following access control (AC) models: Blacklists, Mandatory AC (MAC), Discretionary AC (DAC), Role-Based AC (RBAC), and Attribute-Based AC (ABAC). This section concludes with a set of recommendations for an AC models suitable for smart product scenarios. It concludes that ABAC models together with Blacklists is the most suitable solution for such scenarios.

The role of AC mechanisms, which are implemented after AC models, is to ensure that only authorized entities are able to access the information and functions of a computer system (principle of authorization) [5].

### A. Blacklist

A Blacklist AC is a very simple AC that blocks all requests from entities that are included in a Blacklist. It is used to thwart known or recurrent attackers. Blacklists have to be configured manually or, sometimes, they can be updated automatically according to to predefined rules, e.g., multiple unauthorized requests, or a series of failed authentication procedures. Blacklists usually outperform other AC mechanisms because their complexity class is lower than those, and its performance can be $\mathcal{O}(1)$ with a very small constant factor for the blacklist lookup. Blacklists are a rather simple to use AC, but also rather inflexible, since there no conditional access policies can be defined.

### B. MAC/DAC

MAC and DAC are two early AC models [6]. MAC and DAC can be seen as complementary approaches, but both link access rights directly to the related entities.

In MAC, a central administrator controls the access rights of each entity of the system. No other entity is able to change the access rights. In such a context, MultiLevel Security (MLS) (such as Bell-La Padula [7]) is an often used approach. In MLS, each entity or object of the system has a security level given by a central authority. Each entity is only able to access other entities or objects that have the same or a lower security levels. Mandatory Integrity Control (MIC) is a similar approach and is used in Microsoft Windows Vista (and later). Processes can only write or delete other objects with an security level lower or equal to their own.

DAC differs from these approaches as each entity can hand its rights over to other entities. That way, users are able to share objects among each other. DAC is used in UNIX and Windows-based systems for sharing data and resources.

### C. RBAC

RBAC [8] introduced a new way by setting roles between the entity and the related rights. That way, each entity can have several roles and each role can be held by multiple entities. For administrative purposes, roles are established first, and afterwards they are assigned to entities. Since roles usually rarely change, this reduces the complexity for administrating RBAC significantly after the first setup. If only those entities change that inherit a role, this can be simply addressed by adding or deleting entities (in form of the name or a unique identifier) that are associated with the regarding role. Roles can change dynamically and in that way the user might gain and lose roles automatically when doing special tasks.

### D. ABAC

One of the newest models is ABAC [9]. ABAC uses attributes instead of roles to link rights to entities. This procedure allows the use of dynamic conditions encoded in attributes, such as the location of an entity, to decide whether to grant access or not. Since the role as well as the security level of an entity can be seen as an attribute, it is possible to integrate concepts known from other AC models like DAC or RBAC.

### E. Hybrid approaches

In reality, the distinction between different AC models is not as strict as shown in this section. There are hybrid models like the Location-Aware Role-Based Access Control (LRBAC) [10], which allows the use of a geographical location as a "role". It is often possible to derive a less complex AC model from a more complex one, e.g., it is possible to create an MAC mechanism from an ABAC model.

### F. Access Control for Smart Products

Smart products are user adaptive devices which require AC mechanisms with maximum flexibility since they are related to the everyday life of a heterogeneous set of end-users. Smart products need to maintain user profiles that have attributes and values about users, such as preferences to fulfill their tasks. ABAC models are an evident candidate for building up AC mechanisms for smart products because they provide maximum flexibility in comparison to the aforementioned AC models.

 ISBN: 978-1-61208-109-0 

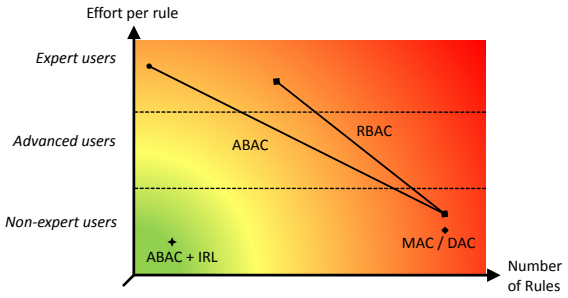



Figure 2. Theoretical comparison of different AC models.

ABAC models are, however, more complex than the other models listed in this section. Such complexity results in a larger consumption of computational resources than simpler approaches. Thus, to reduce to costs of AC operations a Blacklist AC mechanism can be executed before the ABAC mechanism. The Blacklist filters out known misbehaving entities, and their requests do not reach the ABAC mechanism. For instance, after an entity, that was not blacklisted at first, has multiple identical requests denied by the ABAC mechanism, such an entity can be temporarily or permanently added to the blacklist.

AC mechanisms like ABAC are dynamic and flexible. However, they are also hard to configure in the right way. While MAC and DAC have only one way to link access rights to the user, RBAC and, especially, ABAC allows for different ways of connecting access rights to entities through indirect mapping. This flexibility enables very compact and meaningful policy sets. However, if not correctly used, it can lead to an heterogeneous and incomprehensible set of rules. This problem is very likely to occur in case of inexperienced users. This is an important challenge that is addressed with Interactive Rule Learning in Section VI.

The relation between flexibility of an AC mechanisms and the usability is shown in Figure 2. This figure shows that MAC / DAC can be used by non-expert users but the number of needed rules for non-trivial scenarios is extremely high. The figure also illustrates that RBAC and ABAC can have very short rule sets, however, only expert users might be able to do so (since it is difficult to manually define a minimal rule set for a complex scenario). If more rules are used in RBAC and ABAC, it is possible to emulate MAC / DAC mechanisms with the difference that always a role or an attribute is in between entities and their related access rights. Finally, the figure shows that ABAC plus Interactive Rule Learning can be used to create reduced rule sets even by non-expert users.

After defining a suitable AC model for smart products it is still fundamental to define how the rules for such AC model are generated. Such rule generation should consider a set of requirements that are discussed in the next section.

## V. Requirements for AC Rule Sets

In this section, we define the requirements for AC rule sets for smart product scenarios taking into account both security and usability constraints. Not all rules presented in this section are orthogonal, thus conflicts do exist. Such conflicts are detailed and explained in the end of this section.

### A. Security Constraints

The security constraints for building up AC rule sets are regarding specific or permissive rules and also the meaning of such rules. Each requirement is assigned a letter *S* followed by a number.

- *S*1: specific (permissive) rules. Access rules have to be specific enough to leave no opening for intruders. Rules like “everyone is allowed to do everything” render AC mechanisms useless in practice.
- *S*2: meaningful rules. Access rules have to reflect the expectations of the smart product owner. Rules like “every employee of the university is allowed to use the printer” have a better semantic meaning than a similar rule stating that “every one with glasses is allowed to use the printer”, even if every employee of the university wears glasses.

### B. Usability Constraints

The usability constraints for building up AC rule sets are regarding the existence of redundant rules, their consistency and understandability, and also related to the total number of rules. Each usability requirement is assigned a letter *U* followed by a number.

- *U*1: no redundant rules. Rules or set of rules that are fully covered by other rules or set of rules can be deleted without changing the behavior of the AC mechanism. Thus, if a rule set A is a subset of a rule set B, then rule set A can be deleted. Redundant rules only increases the complexity of a rule set without adding any security features and make such sets more confusing for the end user.
- *U*2: consistent rules. Consistent rules mean that two or more different rules must not be contradictory. Contradictory rules could lead to unpredictable access decisions or worsen the usability by unnecessarily increasing the complexity of the rule set.
- *U*3: general, understandable and manageable rule sets. AC rules need to be general enough for users to understand and manage.
- *U*4: minimum number of rules. The number of rules that describes the scenario should be minimal to make the rule set understandable and manageable.

The use of general rules in requirement *U*3 contradicts the requirement *S*1 regarding specific rules. Thus, the best compromise between specific and general rules need to be reached. The best compromise is, however, connected to the users preferences and it is, therefore, individual.

 ISBN: 978-1-61208-109-0 

Rule $U2$ is not only a usability requirement, since it can also impact the security level obtained by the AC mechanism. An inconsistent rule set can lead to a non-expected behavior that can compromise the security of the smart product.

In the next section we develop a rule generation procedure that takes the aforementioned requirements into account. Such procedure combines automatic rule generation with user interaction.

## VI. Rule Generation

Nowadays, the common procedure for rule generation is to do it manually. Therefore, the requirements listed in Section V need to be considered by the owner of the smart product. The manually generation of rules by inexperienced users will likely result in misconfigured access rule sets (or the manual deactivation of security mechanisms), which eventually end up into security vulnerabilities. Therefore, the rule generation process should be automated as much as possible. Learning algorithms, from the Artificial Intelligence research field, are able to accomplish this goal [11].

### A. Automatic Rule Learning

Extracting knowledge out of data by using a rule-learning algorithm is a well-known topic. However, for defining good access rules, a fully automated rule generation is unfortunately not worth most of the time. It is very difficult to determine automatically what kind of information needs to be protected. The whereabouts of a person, for instance. Taxi drivers may have their geographical position public available, but for lawyers or doctors on their way to clients or patients must keep their location information strictly private.

It could be possible to decide which information should be public and private by analyzing the user profile. Thus, automatic rule set generation is possible, but it is expected that errors would also be a commonplace. However, if related information for automatic rule generation is missing, automatic processes are not possible. Hence, the smart product owners have to decide by their own regarding the access rules.

Therefore, a proper solution is to use automatic rule generation to create an initial rule set that is later presented to the user. Such a solution is presented in Section VI-B.

### B. Interactive Rule Learning

Learning algorithms can generate a set of rule sets and present them to users that decide which specific rule set suits best to their context. A rule learner can be used to analyze the set of access rules of a smart product regarding the actual behavior of entities [12].

Such an analysis disclose whether rules are shadowed, redundant, or correlative, and which exceptions exist following the definition and classification presented in [13]. Furthermore, in interaction with users, the number of rules is minimized by analysing, pruning, and rebuilding the set of access rules. This procedure is called Interactive Rule Learning (IRL) [14].

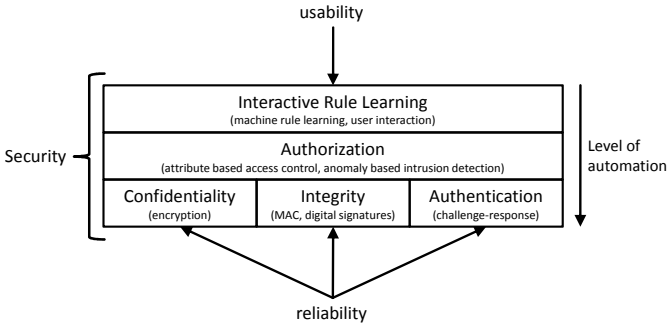



Figure 3. Reliable and Usable CIA + Authorization

Combined with the ABAC, the IRL helps the user to build a secure and usable set of access rules. The expected outcome of ABAC+IRL is shown in Figure 3. This concept represents an important step towards usable security.

An automated rule learning algorithm can fulfill the following requirements: $S1$, $U1$, $U2$ and $U4$. Users have to verify the compliance of requirement $S2$, since it depends on the context and also on the smart product owner preferences. To satisfy requirement $U3$, regarding general rules, interaction between the smart product owner and the rule leaner is required.

## VII. Related Work

Over the years, a variety of learning algorithms have been developed that try to imitate natural learning or use a more technical approach as a starting point. Some approaches try to reproduce the functioning of a brain at the level of neurons [15], [16]. Other mechanisms, such as support vector machines, are based on a more abstract mathematical concept by finding an optimal border between positive and negative examples (like access and deny for an access request) by maximizing the distance between them [17]. Existing algorithms further differ with respect to their applicability, speed, and accuracy [18], [19].

Rule learners use a very intuitive approach in relation to the aforementioned algorithms. They try to find causalities in recorded databases and express them with simple rules. For example, in a database that describes the attributes of different animals like ravens, sparrows and pigs such a rule could be as follows: “If an animal can fly and has feathers, it is a bird”. This approach has the particular advantage of being relatively easy to understand for humans as opposed to the classification of a support vector machine, for instance. This is both a psychological and a practical advantage. From a psychological perspective people tend to accept something more likely if they are able to understand it. From a practical point of view, potential errors can be more easily detected and extended [14].

 ISBN: 978-1-61208-109-0 

## VIII. USE CASE SCENARIO

In our use case scenario we consider a family of four. Alice ($A$), Bob ($B$), and their children Charlie ($C$), a 17-year old, and Denise ($D$), an 8-year old. The set with elements $\{A, B, C, D\}$ is the family, and the subset with elements $\{A, B\}$ are the parents. In the family's kitchen there are 3 new smart products: a smart coffee machine ($X$), a smart blender ($Y$), and a smart oven ($Z$).

We assume that newly bought devices come with a default set of access rules, which are defined by the smart product manufacturers. Since the manufactures cannot predict in which way smart products are going to be used, the factory settings for the access control rules are basically general. They follow the usage rules of similar non-smart products, i.e., everyone that physically interacts with a device is allowed to use up to its full-functionality. For instance, everyone locally interacting with the coffee machine is allowed to brew coffee.

The full control of a smart product is given to the one who first activates it. A smart product might be remotely controlled by its users (through smart devices) after it has been integrated into the home environment. *A* wants to configure and generates access rules for the 3 newly bought smart products (*X*, *Y*, *Z*), so that her family can best profit from them. Three classes of access rights are preloaded in smart devices (those classes can later be reconfigured or changed):

1) *Full access*: the right to locally or remotely access a smart product and to manage its access rights.
2) *Remote and local access*: the right to locally and remotely access a smart product.
3) *Local access*: the right to locally access the smart products.

*A* wants to grant *B* with full control over all the smart products. *C* shall get access to the full functionality (locally and remotely), but shall not have administrative rights over the smart products. *D* shall not have any access to the devices, even by local interaction. Since the family often have guests, *A* wants them to be able to locally interact with the smart products, just as in a non-smart kitchen. The initial manually generated rule set is:

```
Rule Set 1
 1: If (owner) -> full access
 2: If (any)   -> local access
 3: If (A)     -> full access
 4: If (B)     -> full access
 5: If (C)     -> remote and local access
 6: If (D)     -> no access to X
 7: If (D)     -> no access to Y
 8: If (A)     -> no access to Z
 9: If (guest) -> local access
```

There are a few mistakes in *A*'s manually generated rule set. They are:

- The first two rules are residues from the preloaded factory default rule set. The fact that *A* ignored them leads to two implications regarding requirements $U1$ and $U2$. Rule 2 is a superset of rule 9, and it also contradicts rules 6, 7, and 8. Moreover, since there are redundant rules, their number is surely not minimum, which contradicts $U4$.
- Rule 9 is misconfigured since it does not reflect *A*'s expectation. Instead of denying *D*, she denied herself to access *Z*. It contradicts requirements $S2$ and $U2$.
- The rules were generated taking into account specific family members instead of more general attributes, such as age. The use of attributes for generating small and understandable rule sets is recommended and one of the reasons why ABAC is better suited for smart products, as mentioned in Section IV. Therefore, there is a contradiction with $U3$.

The smart products analyze the manually generated rule set taking the usability and security constraints presented in Section V and produce new rule sets that are free of conflicts. In our example, the smart products present to the user *A* two automatically generated rule set options:

```
Rule Set 2
 1: If (age > 40) -> full access
 2: If (family & age > 16) -> remote and local acc
 3: If (age > 9)  -> local access
```

and,

```
Rule Set 3
 1: If (parents)  -> full access
 2: If (family & age > 16) -> remote and local acc
 3: If (age > 16) -> local access
```

It is up to *A* to decide which rule set suits her needs the best. Both rule sets look much better and concise than the manually generated rule set. However, the first rule of the Rule Set 2 is way too general (an infringement to requirement $S1$), since it gives full access rights for everyone above 40, which would include eventual guests. The last rule of Rule Set 2 is also not of her likes, since *A* would not trust a 9-year old to operate kitchen appliances (but she would trust a 12-year old). Thus, *A* picks Rule Set 3, but manually changes rules 2 and 3 to better fits her expectations. The modified rule set, Rule Set 4, is:

```
Rule Set 4
 1: If (parents) -> full access
 2: If (family & age > 12) -> remote and local acc
 3: If (age > 12) -> local access
```

A comparison between the manually generated Rule Set 1 and the interactive generated Rule Set 4 demonstrates a great improvement of the latter regarding the usability and security requirements presented in Section V. Rule Set 4 addresses the security requirements $S1$ and $S2$ since the rules are specific and meaningful. Usability requirements $U1$, $U2$, $U3$, and $U4$ are also fulfilled since there are no redundant rules, and the rules are consistent, understandable, meaningful, manageable and provide a minimum amount of rules to express the owner's security expectations.

 ISBN: 978-1-61208-109-0 

Table I
USABILITY OVERVIEW TO CIA AND AUTHORIZATION.

| | Confidentiality | Integrity | Authenticity | Authorization |
|---|---|---|---|---|
| **Usability** | Yes, transparent | Yes, transparent | Yes, transparent | Partially, fully automation not possible |
| **Adequate Method** | Encryption | Digital Signatures | Proofs of knowledge, biometric traits or digital tokens + public-key enc. | ABAC and Interactive Rule Learning |

## IX. CONCLUSION

In this paper we showed that access control rule sets is the most challenging aspect for combining usability and security in smart product scenarios. Other security services, such as confidentiality, integrity, and authenticity can be automated and, therefore, made fully transparent for end-users. In Table I we summarize the usability aspects and security mechanisms in regarding to the aforementioned security services.

Based on analysis of the different AC mechanisms the combination of a blacklist with an attribute based approach is proposed to fulfill todays and future needs for adaptive devices. We listed a series of security and usability requirements for access control rule sets. We showed that the combination of automated rule learning with user interaction is able to fulfill such requirements to a secure and usable system.

Future work is going to exploit how interactive access rule learning for ABAC can be used to achieve the best results. Initially, data needs to be collected for the automated rule generation from two different sources. The first data source is composed of rules that already pre-loaded or added by users to smart products. The second source is the actual behavior of users of smart products. The combined data is used by the automatic rule learner to define a new set of rules that are submitted to the user for approval.

In the future we are also going to address the processing of hierarchical data in automatic rule learning. Rule learning on hierarchical data is important to allow the users to define natural access rules. Hierarchical data provides contextual information. They are commonplace in many aspects of our daily lives. For instance, business structures are mostly hierarchical, with directors, managers, and secretaries. Current automatic rule learners are not able to process hierarchical data and, therefore, they need to be extended to accept such data[2]. A final aspect is the conversion of automated generated rules to rules that are user-friendly.

## ACKNOWLEDGEMENTS

This research paper is part of research conducted by the EU project SmartProducts, funded as part of the $7^{th}$ framework program of the EU (grant no. 231204) and supported by the Center for Advanced Security Research Darmstadt (CASED).

[2]There are indeed already proposals of rule learning on hierarchical data [20]. However, those are still very limited regarding the use of the hierarchical structures.

## REFERENCES

[1] M. Beckerle, "Towards Smart Security for Smart Products," in *AmI-Blocks'09: 3rd European Workshop on Smart Products*, 2009.

[2] L. Cranor and S. Garfinkel, *Security and Usability*. O'Reilly Media, Inc., 2005.

[3] V. Reding, *The Future of the Internet - A conference held under the Czech Presidency of the EU*. Belgium: European Commission - Information Society and Media, 2009, ch. What policies to make it happen?, pp. 2–5.

[4] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication," 1997.

[5] F. Stajano, *Security for ubiquitous computing*. John Wiley and Sons, 2002.

[6] S. Brand, "DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria (Orange Book)," *National Computer Security Center*, 1985.

[7] D. Bell and L. La Padula, "Secure computer system: Unified exposition and Multics interpretation," *MTR-2997*, 1976.

[8] D. Ferraiolo, D. Kuhn, and R. Chandramouli, *Role-based access control*. Artech House Publishers, 2003.

[9] E. Yuan and J. Tong, "Attributed based access control (ABAC) for Web services," in *IEEE International Conference on Web Services ICWS 2005. Proceedings*, 2005.

[10] I. Ray, M. Kumar, and L. Yu, *LRBAC: A location-aware role-based access control model*. Springer, 2006.

[11] J. Carbonell, R. Michalski, and T. Mitchell, *An overview of machine learning*. Tioga Publishing Company, Palo Alto, 1983.

[12] J. Fuernkranz, "Separate-and-conquer rule learning," *Artificial Intelligence Review*, vol. 13, no. 1, pp. 3–54, 1999.

[13] H. Hamed and E. Al-Shaer, "Taxonomy of conflicts in network security policies," *IEEE Communications Magazine*, vol. 44, no. 3, pp. 134–141, 2006.

[14] M. Beckerle, "Interaktives Regellernen," Master Thesis, Technische Universität Darmstadt, 2009.

[15] M. Riedmiller, "Advanced supervised learning in multi-layer perceptrons-from backpropagation to adaptive learning algorithms," *Computer Standards and Interfaces*, vol. 16, pp. 265–278, 1994.

[16] E. Yair and A. Gersho, "The Boltzmann perceptron network: A soft classifier," *Neural networks*, vol. 3, no. 2, pp. 203–221, 1990.

[17] B. Schoelkopf, C. Burges, and A. Smola, *Introduction to support vector learning*. MIT Press Cambridge, MA, USA, 1999.

[18] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, 2005.

[19] S. Haykin, *Neural networks: a comprehensive foundation*, 3rd ed. Prentice Hall, 2008.

[20] W. Cohen, "Fast effective rule induction," in *Proceedings of the Twelfth International Conference on Machine Learning*, 1995, pp. 115–123.

 ISBN: 978-1-61208-109-0 

# Incremental Online Evolution and Adaptation of Neural Networks for Robot Control in Dynamic Environments

Florian Schlachter[†*], Christopher S. F. Schwarzer[‡*], Serge Kernbach[†], Nico K. Michiels[‡], and Paul Levi[†]
[†]*Institute of Parallel and Distributed Systems*
*University of Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany*
*Email: {Florian.Schlachter, Serge.Kernbach, Paul.Levi}@ipvs.uni-stuttgart.de*
[‡]*Institute for Evolution and Ecology*
*University of Tübingen, Auf der Morgenstelle 28, 72076 Tübingen, Germany*
*Email: {Christopher.Schwarzer,Nico.Michiels}@uni-tuebingen.de*
[*]*These authors contributed equally.*

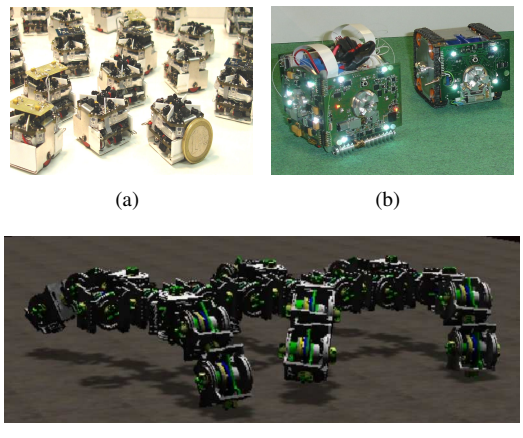
***Abstract*—Many approaches have been developed to tackle the design complexity of modern robotic systems by using evolutionary processes. Starting with an initial solution, the evolutionary process tries to adapt to a given scenario and in the end produces an improved solution. Previous work showed that incremental evolution, a stepwise increase in the scenario difficulty, can increase the success of evolutionary adaptation. In this work, we clearly confirm this effect in the context of online evolution of neural networks. The goal of our online evolutionary approach is to produce on average good, intermediate solutions while the system is adapting. We show that also the average performance of the continuous evaluations is increased by evolving first in a simple scenario and then transitioning to a more difficult scenario.**

***Keywords*-online evolution; neural networks; robotics;**

## I. Introduction and Background

In evolutionary robotics, the design of the robot controllers is driven by bio-inspired approaches [1], [2], [3]. Many of them are evolved offline on an external computer. After optimizing the controllers for a certain task, the best controllers are deployed to the robots. For the evolution of robot control, neural networks play an important role hence to their close relationship to natural systems. In several approaches it has been shown, that the evolution of neural networks can be speed up, by structural evolution of the networks. One of the early works in this field is the Generalized Acquisition of Recurrent Links (GNARL)[4]. In this work, they developed algorithms for the evolution of neural networks with recurrent links. The networks are randomly initialized (random hidden neurons and links) and evaluated. Afterwards, fifty percent of the population are allowed to create offspring (two children) for the next generation and so on. In the NeuroEvolution of Augmenting Topologies (NEAT)[5] the structural evolution starts with empty neural networks and develops over time. They also introduced a cross-over mechanism based on historic information and showed mechanisms for innovation protection (speciation). The improvements to the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) [6] extend the algorithms with a generative encoding and inclusion of sensors and output geometries [7].

Since robots operate in real world, the environment and conditions are subject to continuous changes. Through interaction and disturbances by other robots, humans or changes in the environment, control structures or functions can be obsolete or improper for the current task and need further adoption. Especially, in dynamic scenarios, the requirements to fulfil a defined task (implicitly defined in the fitness function) are subject to changes. Often this changes are hard to predict and occur randomly. One way to deal with this is a continuous process of adaptation of the robot controller to fit to the environment and requirements. This process of adaptation has to be performed on the robots during runtime, since the necessary changes are not known in advance. So the robot needs to evaluate its performance and an integrated evolutionary engine drives the evolution and thus the online adaptation. Additionally, this process can be embedded into an incremental evolution. The advantages of incremental



(a) (b)

(c)

Figure 1. (a) Jasmine swarm (b) Prototypes of the Symbrion and Replicator Robots (c) Exemplary Organism in the Symbrion and Replicator Simulation.

 ISBN: 978-1-61208-109-0 

evolution were also proved by Gomez and Miikkulainen for a prey capture scenario [8] and by Barlow [9], where the complexity of the scenario for an aerial vehicle grows over time and the controller can develop step-wise.

The main goal of our work with evolutionary robotics is to create a system that is capable of adapting controllers online with the necessary complexity for controlling symbiotic robotic organisms [10]. This is a major part of the grand challenges of the Symbrion and Replicator projects (www.symbrion.eu and www.replicators.eu) both for swarm robots and artificial organisms like depicted in Figure 1.

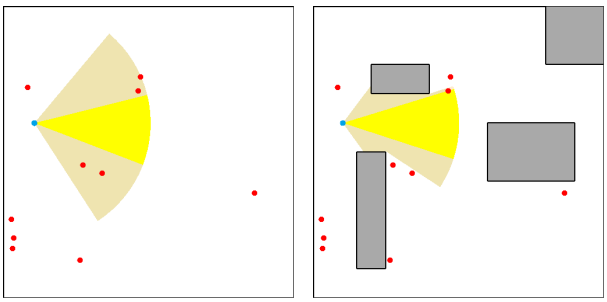
The paper is organized in the following way. In Section II we introduce our approach for evolutionary design of robot controllers and enlighten the different aspects of ongoing work and performed experiments. In the following Section III we present the results of the applied experiments and their impact to our work and finally we conclude the paper in Section IV.

## II. Experimental Setup and Implementation

### A. Arena and Experimental Setup

We evaluated our approach for simulated online evolution in a multi-agent simulation framework that uses a 2D physics engine to simulate a virtual environment. The robot is modelled as an agent in a two dimensional square arena with a size of 500x500 units that is surrounded by impassable walls. Within this arena, there are always 10 red points that symbolize energy sources for the robot, power cubes. These power cubes are static physical objects and pose an obstacle for the robot of the same size as the robot. If the robot is in close proximity of a power cube, it gains one reward point every 50 simulation ticks. After a power cube has dispensed 10 reward points, it is removed and a new, fully charged power cube is placed on a random position in the arena. The sequence of random positions for power cubes is the same for each run. Two arena setups are used; one completely empty and one with four large impassable boxes in a fixed configuration as seen in Figure 2. This particular configuration was chosen to provide a more complex scenario with more obstacles and a differently structured environment. In the empty arena the robot has a red power cube in sight most of the time and can trail a path from cube to cube without having to actively explore the arena. We also considered a maze layout with many thin wall segments scattered in the arena but this promoted simple wall following behaviours which was more simple to adapt to than the empty arena.

The simulated robot is equipped with seven virtual sensors: three sensors to detect the red power cubes in a field of vision with a range of 200 units; three distance sensors with a range of 100 units, in the same layout as the red sensors; one sensor that detects if a power cube is in immediate vicinity. The orientation and location of the red sensors is exemplary shown in Figure 2(a). The yellow cone is the middle sensor, while the light yellow coloured ranges are the left, respectively right sensor. The blue circle represents the robot, the red circles are collectible power resources. The range of the sensors is limited by the walls and obstacles in the arena 2(b). The robot has two actuators that simulate a differential drive with two wheels.



(a) (b)

Figure 2. The two arenas used in the simulated experiment runs: the empty arena **(a)** and the arena with boxes **(b)**. Black lines are impassable walls, red circles are simulated power cubes and the blue circle is the simulated robot. Shown is the initial configuration of the robot and red points which was used for each run. Also shown are the three fields of vision of the sensors of the robot which detect the power cubes.

### B. Neural Network and Control

The robot is controlled by an artificial neural network with recurrent connections and no restriction on network topology. This allows us to find good solutions regarding the complexity of the neural net. The decision of how many hidden layers, connections and neurons are necessary is shifted from human design to an evolutionary automated process. Doing so, the evolution of the neural net can find an optimal balance of the number of neurons and their connectivity. Design decisions made by humans can have no influence to the ability to adapt or can hinder the development of the neural nets by weak start configurations. The network itself performs one update step at each simulation time tick. It has eight input neurons (seven sensors plus bias neuron) and two output neurons. All inputs are mapped to values from 0 to 1, the output neurons provide values from -1 to 1. The values of the two output neuron values is transformed with Equation 1, which gives two positive values $l'$ and $r'$. These modified actuator values are interpreted as a change to bearing $b$ and linear velocity $v$ as seen in Equations 2 and 3. Afterwards, the output is multiplied with a constant factor to scale the values to the simulation and set the velocity or change the bearing directly without simulation of inertia.

$$o' = \frac{o+1}{2} \tag{1}$$

$$b = r' - l' \tag{2}$$

$$v = r' + l' \tag{3}$$

 ISBN: 978-1-61208-109-0 

The described actuator mapping smoothens the fitness landscape for a completely undifferentiated start network because output values of 0 produce a straight forward movement. Note that with this mapping, the network needs to output -1 on both output neurons to come to a full halt and it is impossible to drive backwards. Different actuator mappings had a large impact on the performance of the evolutionary process during preliminary experiments. This particular mapping was chosen as a compromise between a challenge for the evolutionary adaptation and to allow a smooth evolutionary start with an undifferentiated initial network.

### C. Evolutionary Algorithm

For the evolutionary process, we use a genome that encodes the neural network in a structure similar to NEAT [5]. The genome is a list of connection genes and each gene encodes one neural link with source neuron id, destination neuron id and link weight. In the mutation operator, each gene changes its weight with probability 0.2 by applying a uniform random change from -0.2 to 0.2, capped in the range -1 to 1. Additionally, with a probability of 0.4, one structural change is made: Deleting a link, creating a new random link with weight 0 or creating a new hidden neuron by inserting it into an existing link. There is no mutation to delete hidden neurons, however hidden neurons are automatically removed if they are unconnected. For crossover, we are also faced with the problem of finding a suitable mechanism to avoid known problems of recombination of neural networks. The original NEAT approach [5] and likewise the rtNEAT extension [11] is not directly transferable due to a missing supervisor to track the innovations for crossover and due to the small number of robots for speciation and innovation protection.

For the evolutionary engine, we use an evolutionary algorithm based on the $(\mu+1)$ algorithm [12] with a random parent selection and elitism survivor selection scheme and no cross-over operators. The algorithm maintains a population of ten genomes. For each evaluation, one genome is uniform randomly picked to produce one mutated offspring which is evaluated next. After evaluation, the worst in the population is replaced if the evaluatee is better.

To evaluate the performance of the individual robot controller we tried to find an implicit fitness value. Since we can not create new robot offspring, the possibility to measure the performance by reproduction rate is limited. Alternatively, a virtual life energy or power resource can be used. Within the scenario the robots are able to collect power resources. Finally, the robots with a high rate of collected cubes have automatically a high fitness. This includes implicitly the ability of collision avoidance. The robots have to avoid obstacles and drive on optimal paths in order to keep a high movement speed. In case of collision or suboptimal paths, the robot is slowed down or fails to collect the resources.

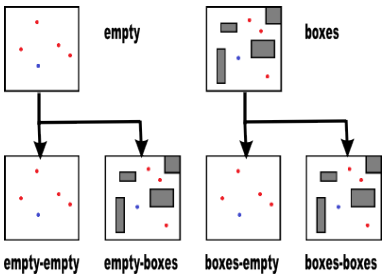



Figure 3. The six different treatments of the experimental setup. In the first two treatments, *empty* and *boxes*, a basic population of undifferentiated networks is evolved to adapt to their respective arena. In the second set of treatments the evolved populations are redeployed and evolved in the other arena (treatments *empty-boxes* and *boxes-empty*) or in the same arena again (treatments *empty-empty* and *boxes-boxes*).

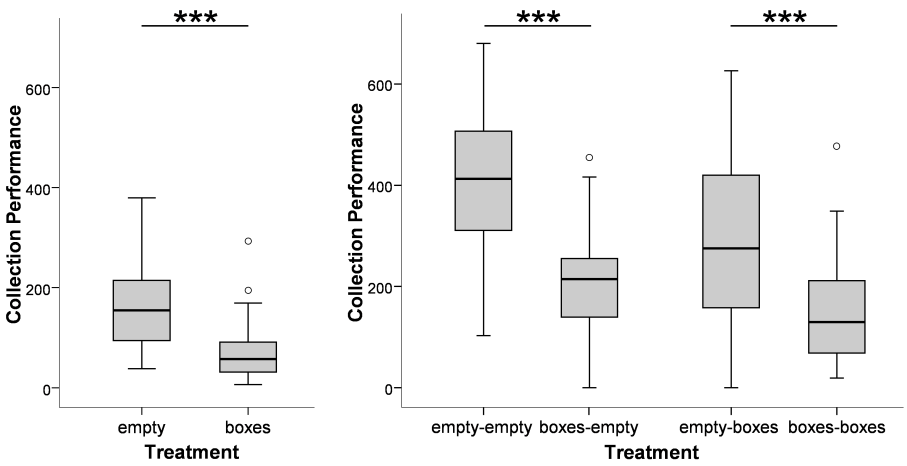
### D. Experiments

In one treatment, we let the robots evolve in an empty arena for 100 evaluations (*empty* treatment). Afterwards we placed these controllers in the same arena for another 100 evaluations (*empty-empty* treatment). Additionally, we placed the same controllers in the arena with obstacles (*empty-boxes* treatment). The motivation of changing the arena is to simulate unforeseen changes in the environment. A preevolved controller is suddenly faced with a new situation. In the empty arena, a controller implicitly avoids obstacles, as long as it can see any red power cube to follow. The chance to see power cubes is minimized in the second arena and the controller has to advance the ability for exploration. The fitness function was always the same. Each robot was awarded for collecting the power cubes. Possible collisions are implicitly punished by slowing down the robot.

The initial population of treatments *empty* and *boxes* is a genome for a perceptron without hidden neurons. There are links from each input neuron to each output neuron and each links' initial weight is 0. At each run, the robot is placed in the same starting position with 10 power cubes placed in the same initial configuration. Each run lasts 100 evaluations and each evaluation is done for 5000 simulation ticks. No changes to the arena and agent states is done in between the evaluations to simulate online evolution. Specifically, the robot remains in its position as well as the power cubes.

An overview of the experimental setup is given in Figure 3. After the $100^{th}$ evaluation in treatments *empty* and *boxes*, the final population of each run is stored. These evolved populations are used as starting populations for a second set of treatments. The evolved populations are put into a different arena in treatments *empty-boxes* and *boxes-empty* or put into the same arena again in treatments *empty-empty* and *boxes-boxes*. The runs in this second set of treatments last again 100 evaluations. For each treatment of the second

 ISBN: 978-1-61208-109-0 




(a) (b)

Figure 4. The collection performance of different treatments. The middle bar marks the median, the box marks the lower and upper quartile and the whisker the minimum and maximum values of the 40 replicates. **(a)** Shown is the summed score of the last 10 evaluations of 100. The performance is lower in the arena with boxes (*Wilcoxon Rank Sums test* $n = 40$ $z = -5.2$ $p < 0.0001$) indicating that it is more difficult to collect points than in the empty arena. **(b)** The end performance after 100 more evaluations of the final populations of treatments *empty* and *boxes* in different arenas. The treatments that evolved first in the empty arena show a better final performance both in the empty arena (*Wilcoxon Rank Sums test* $n = 40$ $z = 5.6$ $p < 0.0001$) and in the box arena (*Wilcoxon Rank Sums test* $n = 40$ $z = -3.9$ $p < 0.0001$).

set one different, evolved population of the first set was used for each run. The evolved genomes were not mixed between populations and each evolved genome was only used once per treatment.

## III. Results

The performance of the evolutionary process was measured by summing the collected score in a window of 10 consecutive evaluations. After the first set of treatments of simulated online evolution over 100 evaluations in the empty and boxes arenas, the performance of the last 10 evaluations is shown in Figure 4(a). The evolved controllers were able to collect significantly more power cubes in the empty arena than in the boxes arena. This shows that the robot collects power cubes slower in the arena with boxes. This arena is more difficult, likely because the robots' sensors are blocked by the boxes and because the robot has to manoeuvre more to drive around the boxes.

After the populations have evolved for 100 more evaluations in the second set of treatments, a general increase in collection performance is seen compared to the first set (Figure 4(b)). The evolutionary process did not fully adapt in the first 100 evaluations and the additional time allowed a further optimisation. Surprisingly, the treatments that were first in the empty arena perform better both in the same arena and in the different arena. It was expected that treatments perform better when they evolved the entire time in one arena rather than when the arena was switched in the middle. This can explain that treatment *empty-empty* performs better than *boxes-empty*. However, it is surprising that treatment *empty-boxes* performs better than treatment *boxes-boxes*. Generally, the arena where the population spent their first 100 evaluations in had a much larger impact on the final performance than the arena switch.

In Figure 5 the collection performance in time windows of 10 evaluations is shown over the course of both treatment sets. The values for the second treatment set are appended to the first treatment set to show the continuous development. There are small peaks in all treatments at evaluation 10 and 110 which must be an artefact of the starting phase of the runs. Presumably, in the random positions of the power cubes there are positions that are easier to collect and these are harvested first. In the initial configuration of power cubes there seems to be a high ratio of those "easy" cubes. After the initial phase, the number of "easy" cubes on the field is lower since they are continuously collected faster than the more difficult ones.

It can be seen in the graph that the performance is continuously increasing over time which shows the adaptive nature of the evolutionary process. The maximum score in one of the 10-evaluations windows is 703 in the empty arena and 517 in the boxes arena and thus we assume that the average performance will further increase after the 200 evaluations in our setup. In this graph it is of note that the *boxes* treatment seems almost stagnant and only after more evaluations in the *boxes-boxes* treatment a significant

 ISBN: 978-1-61208-109-0 

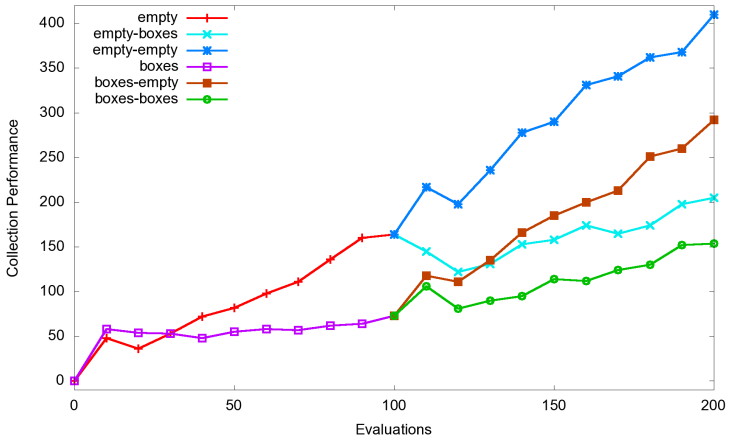
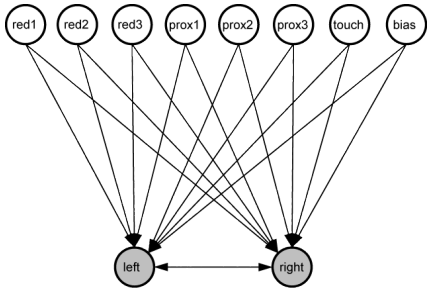
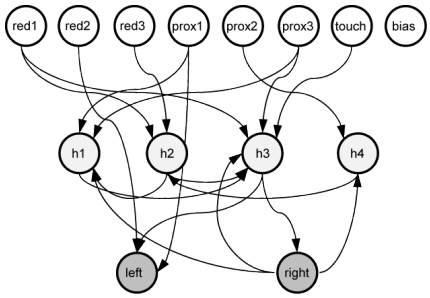



Figure 5. The development of the collection performance over time. Each data point is the summed collection score of a window of 10 evaluations, averaged over 40 runs. After 100 evaluations, the populations were stored and restarted on the same or a different arena. The peaks at 10 and 110 evaluations are an artefact from the initial placement of red points at the start of the runs. The treatment *empty-boxes* is able to maintain some of its advantage of the empty arena in the more difficult box arena. It performs better than the *boxes-boxes* treatment, which had spent more time evolving in this arena.




(a)



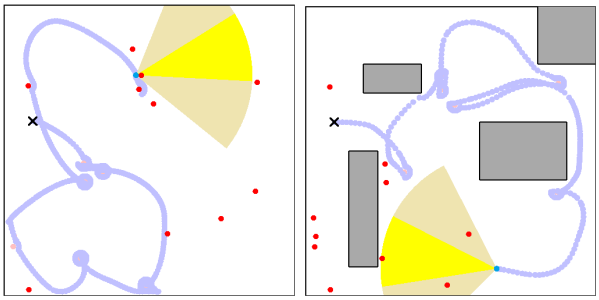

(b)

Figure 6. Two exemplary neural networks from the experiments. **(a)** The initial network with all input neurons connected to the output neurons but with a link weight of zero. **(b)** An evolved, successful neural net for the boxes arena.

upwards slope can be seen. This might explain the bad performance of the treatments that started in the boxes arena because the initial population of undifferentiated networks seems to be very unsuited to evolve efficiently in the boxes arena. In the empty arena on the other hand, the initial population evolves quickly as seen in the much steeper slope of the *empty* treatment. At the switching point of the second treatment set after 100 evaluations, the runs seem to quickly adapt to their new surroundings. The performance growth of the *empty-boxes* and *empty-empty* treatments, as well as *boxes-empty* and *boxes-boxes* treatments are almost the same. In particular, the *boxes-empty* treatment increases its performance faster after the switch from the boxes to the empty arena. Although it is difficult to see due to the aforementioned artefact peaks, the arena switch did not incur a large immediate reduction of performance. The performance of the *empty-boxes* treatment did drop after the switch, but it did not drop below that of the *boxes-boxes* treatment. It seems like the neural networks of the empty arena evolved faster and produced more flexible control structures. These networks had the plasticity to perform well or even better in a different arena compared to the population of networks that were “native” to this arena.

When we take a look at the evolving neural networks, we can clearly see, the structural grow of the networks. Figure 6 depicts the initial network and a exemplary neural network after 200 steps in the empty-box scenario. The top row are the three camera sensors for the red pixels (red1, red2. red3), the proximity sensors (prox1, prox2, prox3), the sensor for touching a food source and an additional bias neuron (not used by this net). The nodes h1, h2, h3, h4 are the evolved hidden neurons and left and right describe the motor output. In Figure 7(a) shows an exemplary run of a robot in the empty arena and Figure 7(b) a more advanced controller in the box arena. In both figures, the view of the sensors and the path of the robot is shown. The cross marks the starting point.

 ISBN: 978-1-61208-109-0 



(a) (b)

Figure 7. Paths of evolved successful controllers for the empty **(a)** and box **(b)** arenas. The cross marks the starting point. The duration of the trace is the same as the evaluation time of the experiment (5000 ticks). Note the tight circling around red points, which is the commonly evolved strategy to stay close to a power cube until it is completely harvested. Only very late controllers evolved that halted in front of the cubes.

## IV. Conclusions

In this paper, we showed the feasibility and advantages of structural online evolution combined with a stepwise increase of the scenario difficulty. We showed ways for structural online and onboard evolution and performed experiments with promising success. It is obvious, that future more complex tasks need a big amount of hidden neurons and recurrent links. The proposed system gives a design tool and automatism at hand, to unburden the developers from the decision of structure and number of neurons.

Regarding incremental evolution, we showed that artificial evolution has different speeds of adaptation depending on the scenario and the initial population. With a given initial evolutionary population and a given target scenario there is a set of intermediate evolutionary scenarios with relaxed difficulty where evolutionary speed is higher than in the target scenario. As seen in our experiments, with the initial population of undifferentiated networks and the target scenario of the boxes arena, the fastest adaptation to the boxes arena was achieved by first evolving in the empty arena and later transitioning to the boxes arena. In this case, the empty arena acted like a relaxed scenario with reduced difficulty than the boxes arena. Skills and structures are quickly evolved in relaxed environments that still give an advantage in different and more difficult environments.

For future work, we want to extend the scenarios with additional robots and a non-supervised mechanism for crossover, so that evolved controllers can be transferred to less evolved robots. Even so, the focus shifts to the transition from robot swarms to artificial organisms and their actual control.

## Acknowledgment

The authors would like to thank Nadine Timmermeyer from the University of Tübingen for her help. The "SYMBRION" project is funded by the European Commission within the work programme "Future and Emergent Technologies Proactive" under the grant agreement no. 216342. The "REPLICATOR" project is funded within the work programme "Cognitive Systems, Interaction, Robotics" under the grant agreement no. 216240. Additionally, we want to thank all members of projects for fruitful discussions.

## References

[1] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology,Intelligence,and Technology*. Cambridge, MA, USA: MIT Press, 2000.

[2] D. Floreano and C. Mattiussi, *Bio-Inspired Artificial Intelligence: Theories, Methods, and Technologies*. The MIT Press, 2008.

[3] D. Floreano and L. Keller, "Evolution of adaptive behaviour in robots by means of darwinian selection," *PLoS Biol*, vol. 8, no. 1, January 2010. [Online]. Available: http://dx.doi.org/10.1371/journal.pbio.1000292

[4] P. J. Angeline, G. M. Saunders, and J. P. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, January 1994.

[5] K. O. Stanley and R. Miikkulainen, "Evolving neural network through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[6] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artif. Life*, vol. 15, no. 2, pp. 185–212, 2009.

[7] D. B. D'Ambrosio and K. O. Stanley, "A novel generative encoding for exploiting neural network sensor and output geometry," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 974–981.

[8] F. Gomez and R. Miikkulainen, "Incremental evolution of complex general behavior," *Adaptive Behavior*, vol. 5, pp. 5–317, 1996.

[9] G. J. Barlow, C. K. Oh, and E. Grant, "Incremental evolution of autonomous controllers for unmanned aerial vehicles using multi-objective genetic programming," in *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems (CIS)*, Singapore, December 2004, pp. 688–693.

[10] P. Levi and S. Kernbach, Eds., *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*. Springer-Verlag, 2010.

[11] K. Stanley, B. Bryant, and R. Miikkulainen, "Real-time neuroevolution in the nero video game," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 6, pp. 653 – 668, dec. 2005.

[12] A. Eiben, E. Haasdijk, and N. Bredeche, "Embodied, online, on-board evolution for autonmous robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, P. Levi and S. Kernbach, Eds. Springer-Verlag, 2010, pp. 367–388.

 ISBN: 978-1-61208-109-0 

# Replica Voting Based Data Collection: Adaptive Protocols and Application Studies

Mohammad Rabby, Kaliappa Ravindran
The City University of New York
New York, NY 10031, USA
Email: mfrabby@yahoo.com; ravi@cs.ccny.cuny.edu

Kevin A. Kwiat
Air Force Research Laboratory
Rome, NY, 13441, USA
Email: kwiatk@rl.af.mil

***Abstract*—**
*Data collection in a hostile environment requires dealing with malicious failures in the sensing devices and underlying transport network: such as data corruptions and message timeliness violations. Functional replication is employed to deal with failures, with voting among the replica devices to deliver a correct data to the end-user. The paper describes the protocol-level design issues for voting with low message overhead and delivery latency under various failure scenarios. The well-known 2-phase voting protocol is employed as a building-block, with different protocol variants dynamically selected based on the network conditions and device-level fault severity. A case study of replicated web services is also presented to illustrate the usefulness of our dynamic adaptation mechanism.*

**Keywords - Majority voting, sensor replication, malicious faults, voter asynchrony, adaptation to environment.**

## I. Introduction

In a distributed information system, it is often required to move the data collected from an external environment to the end-user (e.g., market indices for stock-purchase, terrain images for surveillance). Problems arise however due to failures occurring during data collection, because of the exposure of data collection devices and/or the data flow paths to hostile external conditions. The failures often manifest as data corruptions by malicious devices and timeliness violations in the processing and communication paths.

The devices are often replicated to mask out failures, with some form of majority voting employed to reach agreement on the data fielded by various replicas [6]. In this paper, we focus on the voting protocol mechanisms to deal with data corruptions and enforce a timely delivery of correct data to the end-user. We employ a variant of the 2-phase majority voting to validate the data fielded by replicated devices.

The protocol, in its basic form, requires the sending of consent and dissent votes (YES and NO) by devices about a data value being voted upon, to a central site $B$. See Figure 1. Suppose a data $X(v)$ proposed by voter $v$ is put to vote. Thereupon, a voter $v'$ sends YES or NO message to $B$ based on whether its locally computed data $X(v')$ matches closely with $X(v)$ or not. Based on the YES and NO messages received from $\{v'\}$, $B$ determines if $X(v)$ enjoys a majority consent for delivery to the user. The solicitation of votes from

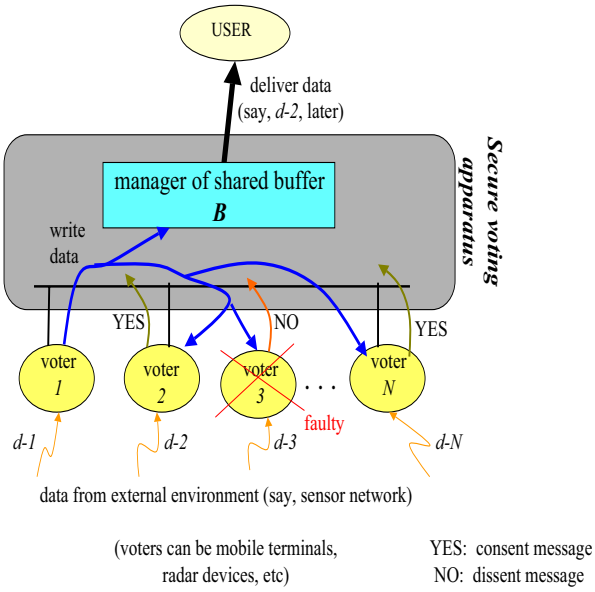



Fig. 1. Distributed voting protocol structure

devices gets repeated until at least $(f_m + 1)$ consent votes are received — where $f_m$ is the maximum number of devices that can be faulty. With $N$ replica devices (where $N \geq 3$), we have $1 \leq f_m < \lceil \frac{N}{2} \rceil$. When $f_m \ll \frac{N}{2}$, the decision on delivering a correct data can be made with less message overhead.

Replica voting has been studied by many researchers in the area of database, file, and web servers: primarily to achieve service-level fault-tolerance and performance. In the context of sensor-based embedded systems however, two important issues come to the forefront. First, the limited battery power of sensor devices imposes a severe constraint on the message-level overhead of voting protocols and the data exchanges between voter devices. Second, the widely varying channel characteristics of communication paths between the devices and the end-user (such as packet loss) can make protocol performance unpredictable. Given these issues, it is necessary that voting protocols are adaptive to the changing operating conditions, while providing a dependable data delivery to the end-user with minimum power and bandwidth consumptions.

The paper describes our extensions to the voting protocol mechanisms to deal with the various failures occuring in the

 ISBN: 978-1-61208-109-0 

environment (such as device attacks and network packet loss). Our extended protocol strives to reduce the number of control messages and application data exchanged between various nodes during voting. This becomes important when large-sized sensor data such as images are considered.

The paper is organized as follows. Section II gives a data-oriented view of replica voting. Section III describes 2-phase commit protocols for voting in resource-scarce systems, with adaptation to the environment conditions. Section IV describes a case study of voting protocols in replicated web services. Section V concludes the paper.

## II. Data-oriented view of voting protocols

In this section, we describe why a content-dependant notion of device faults is necessary in sensor networks, and how this notion impacts the replica voting mechanisms.

### A. Timeliness and accuracy of data

The data delivered to the user as representing an external event (or phenonmenon) may be associated with a timeliness parameter $\Delta$. It pertains to how soon the data produced by a sensor device be delivered at the user since the occurrence of external event represented by this data (i.e., life-time of data). The $\Delta$ parameter impacts the data delivery mechanisms embodied in the voting protocol.

The data generated by a sensor device may be somewhat inaccurate in content (relative to the actual reference datum) due to the inherent sampling errors in the sensing mechanism and/or resource limitations on the data pre-processing algorithms in the device (such as CPU cycles and memory sizes). Accordingly, the bit-level representations of data generated by two different devices may not exactly match — even though the semantic contents of these data may be close enough that they can be declared as representing the same object (as is the case with non-numeric data such as camera images).

Consider, for example, the detection of an enemy plane flying at azimuthal location, say, $35.0^0$. A radar unit may report detection at, say, $35.1^0$ azimuth due to sampling error. Despite this difference in the sampled value and the consequent mismatch in its syntactic representation, the 'data comparison' procedure should treat the two location reports (provided by different devices) as being the same in terms of their semantic contents. The voting system should tackle the computational complexity involved in such a 'data comparison', and still deliver an accurate location report to the Command Center within a few seconds of the presence of enemy plane.

The data generated by a non-faulty device always satisfies the timeliness and accuracy constraints. Whereas, a faulty device may violate the constraints in an unpredictable manner. In the earlier example of radars, a faulty radar unit may report the location of enemy plane as, say, $48.0^0$, or report a more accurate value of, say, $35.15^0$ but after a couple of minutes. In the presence of such faults, the voting protocol should validate a candidate data for its timeliness and accuracy before delivering it to the user. Figure 2 illustrates how the data

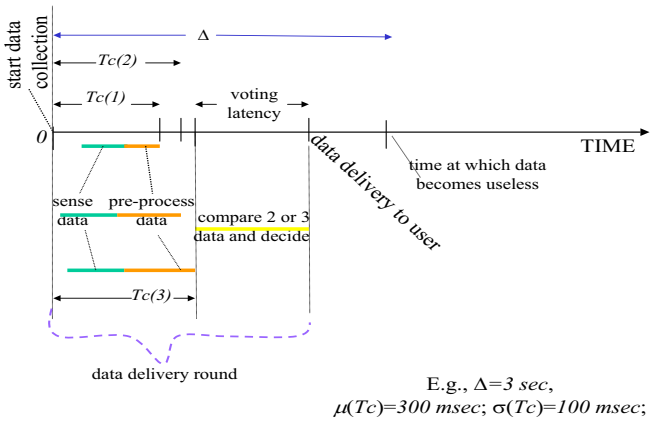



Fig. 2. Illustration of timeliness issues in voting

processing delays incurred for voting impact timeliness at user-level. The device computation time $T_c$ to generate data and the network delay for vote collation are other influential factors.

### B. Protocol-level control of data delivery

From an algorithmic standpoint, the application environment may have at most $f_m$ of the $N$ voters as being faulty, where $N \geq 3$ and $0 < f_m < \lceil \frac{N}{2} \rceil$. This depicts the condition for determining if a candidate data is deliverable to the user.

A functional module $B$ manages a buffer $tbuf$ into which a device writes its data for voting. $B$ resides within the secure enclave of voting machinery, and is securely connected to the user to whom a final result in $tbuf$ gets delivered. The voter devices and $B$ are connected through a secure multicast message channel, where communications are authenticated, have certain minimum bandwidth guarantees, and enforce anonymity among voters. We assume that $B$ is housed within a secure infrastructure that is immune from getting attacked.

A voter first proposes its data by a multicast-write into the remote buffer $tbuf$. From among multiple data items proposed, the buffer manager $B$ selects a candidate data for voting, and then solicits votes on this data. If a majority of consents (i.e., $f_m + 1$ YES votes) are received from the voters, $B$ passes on this data to the user. Otherwise, $B$ selects a next candidate data for voting. If $B$ cannot determine a majority before the data delivery deadline $\Delta$, it discards the data (for safety reasons).

In a real-time system where data may arrive continuously, it is possible that the information loss caused by a missed data delivery can be compensated for by the subsequent data deliveries (e.g., periodic dispatch of terrain maps from a battlefield with adequate frequency). In this setting, the data delivery requirement can be relaxed by prescribing that the rate of missed data deliveries over an observation interval should not exceed a small threshold $\zeta(X)$, where $0.0 < \zeta \ll 1.0$.

### C. Partial synchrony in communications

Partial synchrony in a system means that if an activity starts (say, a network message transmission), it will eventually complete in a finite amount of time. An upper bound on the completion time is however not known to the higher-layer

 ISBN: 978-1-61208-109-0 

algorithm running on the system [9]. In our data collection system, the 'partial synchrony' property manifests as follows.

A non-malicious device will eventually report a correct data (the device is assumed to have adequate computational resources: such as CPU cycles, bandwidth, and battery power). No device can be branded as faulty, in an algorithmic sense, unless it exhibits a sustained bad behavior. A network channel that loses messages intermittantly will eventually transmit a message successfully. And, adequate number of replica units remain in the field so that a management entity assigns the task of data collection to $N$ units for voting purposes.

In the light of above data characteristics and operating environments, we outline the voting protocol extensions next.

## III. Augmentations to voting protocols

In this section, we augment the 2-phase voting protocol with an implicit voting mode to enhance the performance under normal cases, i.e., when packet loss in the network is small.

### A. Explicit determination of majority

2-phase voting protocols require the sending of explicit consent and dissent votes (i.e., YES and NO messages) about a data being voted upon to a central vote collator. The protocol determines a majority based on the number of YES (or NO) votes from among the responses received. That only the votes received are considered in the decision (instead of requiring all the $N$ votes) guarantees liveness in the presence of send-omissions of faulty devices and network message loss [5]. We refer to the protocol as M2PC (modified 2-phase commit).

Note, a non-faulty device $X$ that does not yet have its data locally computed always votes NO for a data $v$ from another voter $X'$ currently being voted upon. This is because $X$ does not have the local context yet to determine if $v$ is a good data or a bad data — and hence $X$ votes NO for safety reason.

The M2PC protocol incurs a worst-case message complexity of $\mathcal{O}(N^c)$, where $c$ is a constant: $1.0 \leq c \leq 2$. The actual value of $c$ depends on the environment, such as the extent of voter asynchrony $\sigma(T_c)$ and the number of faulty voters $f$.

In contrast, a centralized placement of 'data comparison' functions, as would be needed in coordinator based voting schemes [6], requires shipping the large-sized non-numeric data to the central node $B$ (e.g., terrain images from remote cameras). Here, the data movement overhead over the network is: $(f_m + 1)$ in the best case and $(2f_m + 1)$ in the worst case. By careful engineering of the M2PC scheme on the other hand, a voting incurs just 1 multicast data transfer in the best case[1], with the semantics-aware 'data comparisons' carried out locally at the voters in parallel — thereby reducing the drain on battery energy and the voting latency.

### B. 'Voter silence' as consenting vote

The large number of YES/NO message exchanges in M2PC raises scalability concerns with respect to the size of voter complex $N$. An implicit-consent based voting protocol [3] solves the scalability problem. In this protocol, $B$ solicits only dissents, if any, from the voters for a candidate data. The large number of YES responses is avoided by having a consenting voter $X$ not send any message and $B$ treating the 'lack of message from $X$' as an implicit notification of consent. The protocol, referred to as IC-M2PC, rests on the premise that a large majority of voters are non-faulty (i.e., $f \ll \frac{N}{2}|_{N \gg 3}$) and message loss is sporadic in the normal case.

When $f$ is close to $\lceil \frac{N}{2} \rceil - 1$ and/or many dissent messages are lost or excessively delayed, it is possible for a faulty data to be delivered inadvertantly. This is because $B$ construes the non-receipt of a (dissent) message from a voter $X$ as a consent by $X$. The motto is: NO NEWS IS GOOD NEWS — which is an optimistic one ! The likelihood of an undetected mis-delivery of data (due to missed and/or lost dissents) should be kept to a small minimum — say, $< 10^{-2}$. Where a 100% guarantee in the integrity of data delivery is required, a supplementary mechanism should be resorted to by the voting layer to ascertain the voters' intent through other means of low overhead probing. The choice is based on how often the above cases arise and the relative design complexities.

The message overhead of IC-M2PC may be given as $\mathcal{O}(N^{c'})$, where $0.0 < c' \leq 1.0$ in the normal case and $c' < c$ in the worst case (i.e., $f \rightarrow \lceil \frac{N}{2} \rceil - 1$). Here, $c' \rightarrow 0^+$ captures the case of voting on exactly one data item that requires just one message exchange (from the voter who proposed the data). And, $c' \rightarrow c^{(-)}$ captures the case of voting on multiple data items, with all but the last item getting discarded.

Figure 3 illustrates IC-M2PC based voting relative to M2PC, with timing scenarios. As can be seen, the YES vote generated by M2PC does not occur in IC-M2PC. But, IC-M2PC takes longer to complete the voting because of the wait for a full timeout period of $T_w$ to determine if there are any dissents; whereas, M2PC may complete the voting quicker if $f_m + 1$ YES votes are received well before the expiry of timeout $T_w$.

### C. Vote history based sanity check

From an algorithmic standpoint, the network-induced message loss or excessive message delays are indistinguishable from send-omissions of voters. This uncertainty has different implications on M2PC and IC-M2PC: an inability of M2PC to establish a majority consent/dissent if a large number of the voter responses were lost, whereas IC-M2PC may inadvertantly decide to deliver a bad data by believing that the voters whose dissent responses were lost have consented. To deal with this problem, IC-M2PC employs a history vector based mechanism to sanitize the commit decisions.

Here, $B$ solicits the history of how each voter had cast its votes in the last $K$ rounds of voting, where $K > 1$. A voter sends its history information as bit map, with each bit position corresponding to a voting round and indicating whether a YES or NO vote was intended by this voter in the corresponding round. Upon collecting histories from enough number of voters such that a majority determination of the actual votes can be ascertained, $B$ compares its earlier decision for this round. If

[1]M2PC incurs $(f_m + 1)$ data movements only in the worst case. The substantial savings in data movements over the centralized scheme outweighs the $\mathcal{O}(N^c)$ short YES/NO messages needed in the M2PC scheme.

 ISBN: 978-1-61208-109-0 

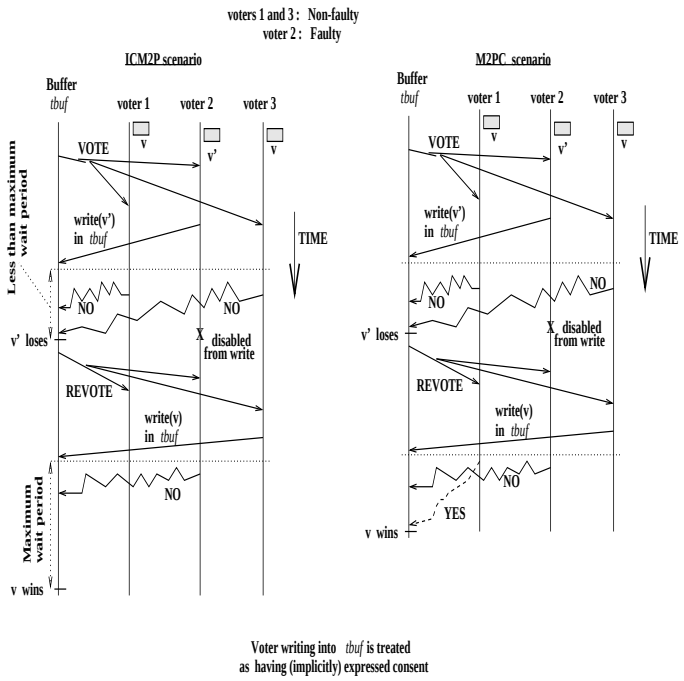



Fig. 3. A voting scenario with M2PC and IC-M2PC protocols

the two decisions match, the result is delivered to the user; otherwise, the result is discarded.

Consider a sample scenario shown in Figure 4. After round 4 in IC-M2PC mode, history vectors are collected, whereupon $B$ finds that the decision in round 2 was erroneously made, and hence caused a discard of the result $d_2$. Since only 1 out of the 4 decisions was erroneous, $B$ attributes the problem as arising from benign message loss conditions. At the end of round 6 however, $B$ finds that both the decisions were erroneously made, and hence suspects a persistent message loss. This causes $B$ to switch to the safer mode: M2PC. That $B$ receives the consent/dissent messages without much loss in rounds 7-9 causes a switch back to the optimal mode: IC-M2PC, in round 10.

We introduce a *resilience* parameter $M$, which is the number of implicit YES votes needed in favor of a data $d'$ (in addition to the YES vote associated with the proposer of $d'$) for $B$ to accept $d'$ as good, where $f_m \leq M \leq (N - f_m - 1)$. Thus[2], if $L$ NO votes are received by $B$ from the remaining $(N-1)$ voters during the dissent timeout period $T_w$, the condition:

$$(N - 1 - L) \geq M \quad (1)$$

determines if the acceptance threshold has been met to declare the data $d'$ as good. $M$ is a tunable parameter in the range $[f_m, N - f_m - 1]$ to control the data miss rate $\zeta$. Choosing a value of $M$ much higher than $f_m$ reduces the occurrence of false negatives, i.e., lowers $\zeta$.

Our voting protocol thus embodies 2 modes of operation: i) the light-weight IC-M2PC that performs well under normal cases when packet loss is small and failures are benign, and ii) the heavy-weight M2PC that is more resilient against severe failures but incurs more message overhead. The protocol evaluates these tradeoffs at run-time, and then switches to the appropriate mode based on the sensed failure conditions.

[2]In M2PC, there is no need for the $M$ parameter, i.e., $f_m$ YES votes (besides the YES from the proposer of $d'$) suffice to accept $d'$ as good.

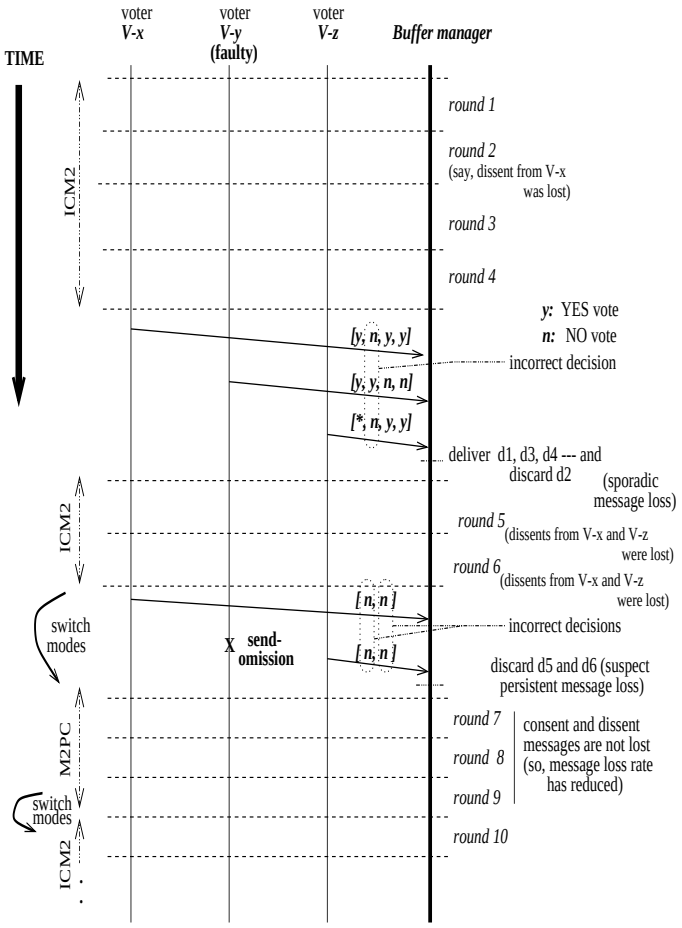



Fig. 4. Vote history to check commit decisions ('*': voter $V_z$ is unaware of round 1)

### D. Voting mode switching control

Since the network message loss rate $l$ may not be directly available to the voting protocol layer, $l$ is determined based on the data misses observed at the interface to the user.

The message loss rate $l$ impacts the data miss rate $\zeta$ in different ways in M2PC and in IC-M2PC. A higher $l$ causes an increase in the duration of data proposal phase due to retransmissions of a candidate data until $B$ and at least $\frac{N}{2}$ voters receive the data. The non-receipt of a candidate data at one or more voters causes them to cast a NO vote for the data (for safety reasons), which makes it possible that no data is delivered to the end-user even after $2f_m + 1$ iterations. Furthermore, the time-to-complete a voting iteration (i.e., decide on the abort or commit of a candidate data) increases in M2PC due to retransmissions of the lost votes for a candidate data. The resulting increase in the time-to-deliver a data may cause more frequent violations of the timeliness

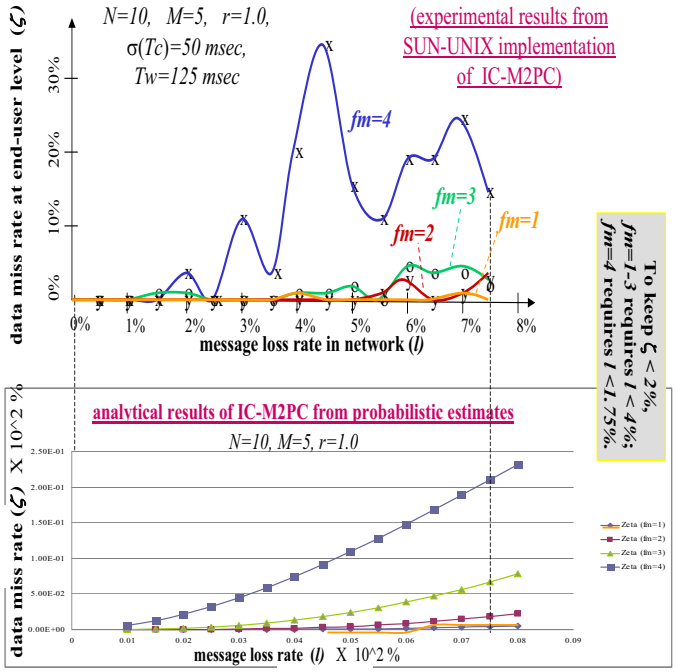



Fig. 5. Experimental and analytical results on data miss in IC-M2PC mode

constraint $\Delta$. In IC-M2PC however, the lost NO votes within a waiting period $T_w$ are treated as consents, leading to a premature tentative commit in favor of a faulty candidate data that subsequently gets invalidated during the history-vector based check. Regardless of how the message-loss phenomenon inter-plays with the data delivery mechanisms in M2PC and IC-M2PC, an increase in $l$ causes an increase in $\zeta$.

The voting layer employs a monitor at the data delivery point to the user to determine the missing data, if any. For this purpose, the monitor agent at $B$ affixes sequence numbers to the voting rounds generated during a IC-M2PC run. At the point of delivery to the end-user, the agent at the user delivery point checks for missing sequence numbers. The observed data miss rate ($\zeta$) allows the monitor to estimate $l$ therefrom.

For the M2PC protocol, the monitor agent at the user end keeps track of the number of missing data in the various voting rounds (using the monitor-assigned sequence numbers). This information, combined with the observed number of missing votes (from the required $N$ votes) in each round during M2PC execution, allows estimating of the message loss rate $l$.

Figure 5 provides the experimental results from IC-M2PC implementation on SUN-UNIX system for the case of [$N = 10, M = 5, r = 1.0$], with $f_m = 1, 2, 3, 4$ and $l = 0.5 - 8\%$. For $f_m = 4$, $\zeta$ increases to beyond 4% for $l \approx 3\%$. For $f_m = 3$, $\zeta$ increases to beyond 3.5% for $l \approx 5\%$. For $f_m = 2$ and $f_m = 1$, $\zeta < 1.5\%$ for $l < 4\%$. These results, combined with the information on voting message overhead and latency, allow the designer to set an acceptable operating region for IC-M2PC vis-a-vis M2PC.

We now describe a case study of replicated web services where M2PC and IC-M2PC protocols can be employed to infuse fault-tolerance during query processing.

## IV. Case Study of Replicated Web-Based Information Service

A web service is provided by one or more App servers that process client queries about the information objects maintained by back-end data servers [11]. The information objects are often structured pieces of pre-processed data about the application's external environment (e.g., data collected from target tracking sensors in a military application, market indices prepared by trend-watchers in a stock market application). The information repository is updated with new objects or object modifications by 'publisher' clients that process raw data from the external environment for writing into the repository [12], [13] (e.g., sensors deployed in a field). 'subscriber' clients query the repository to identify objects of interest via read operations on the data servers. Multiple client queries may be posted concurrently on a web service, wherein the processing actions on these queries share the computational and network resources at the server end.

A web service is more vulnerable to attacks than the back-end data servers that host the web service. This is because of the security weaknesses and diverse usage profiles inherent in the web service operations. Servlet replication is often employed as the basic means to counter the effects of attacks and enhance the service availability. Our goal here is to examine the web service reliability in terms of our probabilistic models in a scenario of servlet replication.

### A. Servlet replication support mechanisms

The servlet replicas are capable of independently processing the queries and computing the results returned to the client. The parameter $f_m$, which depicts the maximum number of replicas that the system designer assumes as fault-prone, strongly impacts the choice of $N$: the total number of replicas. A client query is multicast to all the replicas for processing. The multiple query results are then voted upon to decide on a correct result for delivery to the client. See Figure 6.

The degree of replication $N$ is determined by the cost of replication weighed against the need for a higher reliability of the web service. For a given $N$, the network message-loss phenomenon however impacts the M2PC and IC-M2PC operations in different ways, as measured in terms of voting message overhead and latency.

Voting latency directly impacts the user-level data miss rate $\zeta$ — and hence the query success rate.

### B. Web service reliability: IC-M2PC vs M2PC

The performance measure $\zeta$ is an indicator of *service availability*, depicting how often a client query fails to return a correct result. Here, any attempt to return a faulty result (i.e., a corrupted result or a result that is delayed longer than $\Delta$) is detected by the replica voting protocol.

The candidate output result of a query from a servlet can be decided as correct upon receiving $f_m + 1$ consent votes for

 ISBN: 978-1-61208-109-0 

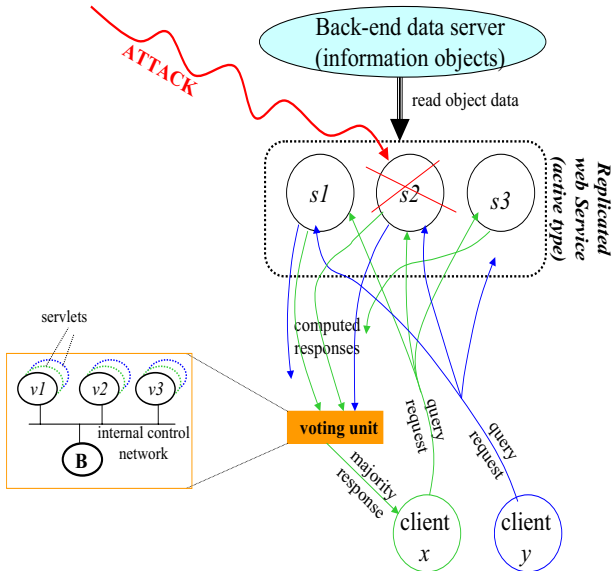



Fig. 6. Servlet replication technqiues for query processing

the result, under the condition: $f \leq f_m$, where $f$ is the actual number of attacked servlets and $1 \leq f_m < \lceil \frac{N}{2} \rceil$ [15].

As for the timely return of a result, the higher latency in voting with IC-M2PC may increase $\zeta$ (and hence lower the content availability), in comparison with M2PC. IC-M2PC however incurs less message overhead (and hence consumes less bandwidth) than M2PC. Furthermore, the protocol designer may conservatively set $f_m = \lceil \frac{N}{2} \rceil - 1$ for a given $N$ (for an increased margin of consent voting) — even though the knowledge about $f$ may instil confidence in the designer to set $f_m$ optimistically to a lower value.

The resilience of a web query is determined by the ability of the web service layer to reconfigure the query processing without exceeding the latency limit $\Delta$ when the servlet currently returning the query result is deemed to have failed. A reconfiguration involves aborting a query result due to lack of enough consent votes and soliciting a new result for the query from a different servlet (i.e., moving to a next iteration in the current round). Since IC-M2PC infers the consent votes for a query result only by implicit means: namely, the non-receipt of dissent votes, a message loss in the network may cause IC-M2PC to initially proceed with incorrect results which then get discarded later by the history-vector mechanism (even if the time elapsed is less than $\Delta$). So, IC-M2PC may incur a higher $\zeta$ (and hence entail a lower resilience) than M2PC when $l$ and/or $f$ is large.

## V. Conclusions

We considered the development of message-optimal voting protocols for data collection in sensor networks. The environment is one of a lossy data transport network, compounded by malicious failures of data collection devices. The functional extensions needed to the basic form of 2-phase majority voting are the provisioning of additional protection layers to sanitize the voting decisions in the presence of sustained message loss. We described two modified versions of 2-phase commit: M2PC and IC-M2PC. M2PC is based on explicit exchange of YES and NO votes; IC-M2PC attempts to reduce the message overhead by suppressing the YES votes. The required augmentations to the protcols to increase their robustness and performance were also described, along with experimental data from a prototype voting system.

The important goal is to ensure the integrity of data delivery to the end-user in the presence of data corruptions and other faults in the sensing system. A main thrust is to reduce the message overhead, while keeping the voting latency to within tolerable limits. Our protocol is highly adaptive to the various types of failures occurring in the data collection environment (such as sensor networks), to meet the goal. We also described a case study of replicated web service employing the voting protocols to provide a fault-tolerant query processing.

## ACKNOWLEDGEMENT

The paper has been approved by AFRL for Public Release (Distribution Unlimited): 88ABW-2010-3936 dated 21 Jul 10.

## References

[1] H. Kopetz and P. Verissmo. 'Real Time Dependability Concepts'. Chap.16, *Distributed Systems*, S. Mullender, Addison-Wesl. Co., 1993.

[2] O. Babaoglu. 'Non-blocking Commit Protocols'. Chap. 7, *Distributed Systems*, ed. S. Mullender, Addison-Wesley Publ. Co., 1993.

[3] B. Hardekopf, K. A. Kwiat, and S. Upadhyaya. 'Secure and Fault-tolerant Voting in Distributed Systems'. In proc. IEEE *Aerospace Conference*, Big Sky (MT), pp. 1117-1126, March 2001.

[4] R. R. Brooks and S. Iyengar. Chap. on 'Sensor Fusion and Approximate agreement'. In *Multisensor Data Fusion*, Prentice-Hall Publ., 1998.

[5] R. D. Prisco, B. Lampson, and N. Lynch. 'Fundamental study: Revisiting the Paxos algorithm'. In *Theoretical Computer Science*, 243:35–91, 2000.

[6] P, Jalote and et al. 'Atomic Actions on Decentralized data'. Chap. 6, *Fault-tolerant Systems*, John-Wiley Publ. Co., 1995.

[7] W. Du, J. Deng, Y.S. Han, and P.K. Varshney. 'A Witness-based Approach for Data Fusion Assurance in Wireless Sensor Networks'. In proc. IEEE-GLOBECOM'03, pp. 1435-1439, Dec.2003.

[8] S. Forrest, A. Somayaji, and D.H. Ackley. 'Building Diverse Computer Systems'. In proc. 6th Workshop on *Hot Topics in Operating Systems* (HotOS-VI), IEEE, ISBN: 0-8186-7834-8, 1997.

[9] M. Castro and B. Liskov. 'Practical Byzantine Fault Tolerance'. In proc. 3rd Symp. on *Operating Systems Design and Implementation*, New Orleans (LA), ACM-DL, ISBN:1-880446-39-1, Febr. 1999.

[10] M. A. Malek, G. Ganger, G. Goodson, M. Reiter and J. Wylie. 'Fault-scalable Byzantine Fault-tolerant Services'. In proc. 20th ACM symp. on *Op. Sys. Principles*, ACM SIGOPS, Brighton (UK), ISBN:1-59593-079-5, Oct. 2005.

[11] S. Ghandeharizadeh, C. Papadopoulos, M. Cai, R. Zhou, and P. Pol. 'NAM: A Network Adaptable Middleware to Enhance Response Time of Web Services'. Chap. II, *Web Services: Researches and Chalenges*, ed. L. J. Zhang, Cybertech Publishing, 2008.

[12] S. Pollack and W. K. McQuay. 'Joint Battlespace Infosphere Applications Using Collaborative Enterprise Environment Technology'. In *proc. SPIE-5820, Defense Transformation and Network Centric Systems*, Ed. Raja Suresh, Bellingham (WA), pp. 235-242, 2005.

[13] IBM Software Group. 'IBM Infosphere Master Data Management Server: Technical Overview'. *White Paper*, Feb. 2008.

[14] A. Sabbir, K. A. Kwiat, K. Ravindran, and M. Rabby. 'Do Centralized and Distributed Voting Methods Complement Each Other ?? A Case Study of Replica Voting Protocol Design'. in *Technical Report*, Dept. of Computer Science, The City University of New York, Sept. 2009.

[15] K. A. Kwiat, K. Ravindran, C. Liu, and A. Sabbir. 'Performance and Correctness Issues in Secure Voting for Distributed Sensor Systems'. in proc. *SPECTS'02*, San Diego (CA), July 2002.

 ISBN: 978-1-61208-109-0 

# Improving Robustness to Environmental Fluctuations – Dynamical Hierarchies Included

Dragana Laketic, Gunnar Tufte
*Department of Computer and Information Science*
*Norwegian University of Science and Technology*
*Trondheim, Norway*
*(draganal,gunnart)@idi.ntnu.no*

***Abstract*—Dynamic environments present an everlasting challenge for any system, be it the one emerging in nature or the one designed by humans. Biological solutions – living systems, successfully tackle the challenge by adapting to their varying environment. Human designers, when faced with a similar challenge, may turn to biology to seek inspiration for possible solutions. Therefore, our previous work has considered strategies and mechanisms for achieving adaptation in living systems. In particular, the focus was on the achievement of adaptation through preservation of homeostasis and the role of hormones within such processes. This paper builds upon those findings. Further, it investigates one possible way to increase the robustness of an adaptive process performed by a man–made system where the system architecture is assumed to be of a grid–of–cells style. We suggest that some of the principles underlying dynamical hierarchies be used to enhance the previously developed model of an adaptive system in order to improve system robustness. This idea is supported with simulations which show that such model exhibits an increased robustness to environmental variations.**

***Keywords*-**

adaptation; environment; dynamical hierarchies; homeostasis; hormones

## I. Introduction

When man–made systems are exposed to dynamical environments, their functionality is endangered. If the system is to preserve its functionality, it needs to adapt. Often, the possibility of human intervention is excluded so besides adaptivity, autonomous fashion of the system operation is desirable, if not required. Also, the robust operation is of particular interest because of the often unpredictable and sometimes harsh nature of environmental variations.

Living systems exhibit these qualities – they preserve their viability in dynamic environments by adapting to environmental variations. Therefore, our investigation has considered adaptive mechanisms and principles in living systems. Previous simulation results have shown how principles of preservation of homeostasis can be used for the achievement of adaptation in a modular man–made system whose modules (cells) are placed in a grid formation [1], [2]. This paper goes a step further and investigates one particular way in which a system can increase the probability of successfully reaching adaptation to environmental variations. It looks into the organisation of living systems and addresses its hierarchical nature examining the effect which the inclusion of principles of dynamical hierarchies into the adaptation process may have on the system robustness to environmental fluctuations.

Engineering applications of the principles of dynamical hierarchies have been recognised as one of the challenges for the topic of *dynamical hierarchies* within the field of Artificial Life [3]. It is our belief that this paper represents a significant contribution to further establishment of this topic by suggesting that the robustness to environmental fluctuations in a man–made system may be increased if the adaptation process performed by the system is enhanced with the principles of dynamical hierarchies.

The paper is organised as follows. Section II gives an introduction on biological principles which were used in our investigation providing the reference to the work of other researchers inspired by similar biological solutions. In particular, the notion of hierarchies within the context of organisation of biological systems is introduced in subsection II-A. Section III briefly presents our previous work. It is included here because the results presented in this paper build up on it. Section IV presents the model of the system under investigation which includes dynamical hierarchies. Further, Section V explains experimental setup, while Section VI provides simulation results. Finally, the conclusion of the presented work as well as some directions for future work may be found in Section VII.

## II. Adaptivity of Living Systems

Adaptivity of living systems is the result of a number of processes which may be viewed as long term and short term processes dependent on the process duration relative to an individual's lifetime. The long term processes refer to the evolution of living systems realised through coevolution with environment. Their effects can be monitored at the population level. Such processes have resulted in individuals possessing inherent adaptive mechanisms. Once the environmental variation is detected, these mechanisms are employed in order to achieve adaptation to new environmental

 ISBN: 978-1-61208-109-0 

conditions. Processes performed by these inherent mechanisms upon sensing environmental fluctuation are short term processes. In general, these processes may be said to aim at the preservation of homeostasis i.e., the steady state of the organism's internal environment despite environmental fluctuations, be it of internal or external origin. A plethora of homeostatic processes is performed by different systems within the body, mutually intertwined and interdependent in their operation [4]. The prominent role belongs to endocrine system which is responsible for control and communication within homeostatic processes.

In some cases, however, environmental variations are such that organisms are unable to adapt despite possessing inherent adaptive mechanisms. Examples may be found at the population level e.g., the extinction of whole species due to abrupt, unpredictable or extreme change, or at the level of one individual when exposed to extreme or harsh senvironmental conditions e.g., too high temperature. Therefore, it can be said that living systems exhibit certain *normativity* in withstanding environmental variations. The higher the probability of withstanding environmental variation, the more robust the living system is to this variation. Further, if the environmental effect on living systems is described by a set of parameters e.g., temperature, pressure, salinity, acidity, then environmental variations are represented by variations in the values of these parameters.

The preservation of homeostasis for adaptive processes as well as the tasks performed by hormones have been addressed by a number of researchers, be it as a general framework for adaptive processes [5]–[7] or for a specific task like control and self–organisation within evolutionary robotics field [8]–[11] or fault tolerance in a multiprocessor system [12]. Our approach aims at establishing a model of an adaptive man–made system inspired by these biological principles which could find its application in adaptive hardware and electronics systems.

### A. Hierarchies

Ever since its creation, the living matter has been faced with a difficult task of preserving the viability despite ever–changing environment. Clever strategies emerged providing optimal solutions not only for the present environmental conditions but also for maintaining the possibility of further evolution. Living systems as we see them today, are the result of such processes. As can be observed, the design strategy which has proven most successful through the evolutionary mechanism of natural selection is a step–wise creation of more and more complex living systems. Their complexity refers to both the complexity of organisation and the complexity of the behaviour they are able to perform, the latter being the result of the former. Different solutions would be created competing with each other and *the winner* would be a stable solution for the present conditions [13].

Further evolutionary processes would include these stable units and create future solutions with these units included as subunits together with some novelty brought about with the resulting, hierarchically higher organisation, as a rule more complex in its organisation and behaviour. Organisms built up of cells, which form tissues, tissues which form organs, organs entwined into systems which make up an organism, represent a well–known example of such organisation. Hierarchical organisation can be noticed even further, towards eco–systems all the way up to the living planet [6]. Such organisation which exhibits hierarchies in structure, where higher level units contain lower level units, also need to account for the dynamics between these entities (units), be it the entities on the same level or entities pertaining to different levels. For the example given in [14], just a mere aggregate of amino acids would not make an enzyme. It takes a certain order of amino acids when forming the enzymatic protein and in addition a certain folding of the protein to yield enzymatic properties. Novel properties of the amino acid sequence which form the enzymatic protein emerge as a result of interactions between amino acids but would be impossible to exist if the hierarchically higher level – the protein, did not possess the novel property i.e., the ability to fold. Observed from the protein level, properties of the pertaining amino acids can be described in a novel way. In general, we speak of levels of description so that a higher description level requires certain amount of information from the lower level to be discarded [13] i.e., some of the dynamics from the lower level need be abstracted – the higher the level in hierarchical organisation, the lower the level of detail with which the pertaining dynamics is described.

Further, organisation into dynamical hierarchies leads to a 'hierarchical control arising from a collection of elements' [15]. In this way, higher levels impose certain control on the lower level dynamics. The novelty brough about by the dynamics at a higher level plays a significant role in generating hierarchical control for the lower level entities. However, dynamical hierarchies still represent a challenge to address which is partly due to the lack of the established theoretical framework for this topic. Although a significant amount of work has been done with the aim of establishing such framework [16]–[19], a number of issues still remain open. Primarily, the criteria for the definition of novelty generated by the system intra- and inter-level dynamics need be established. In [20], the necessary condition for the existence of a new level is given from the information theoretic approach. There, this condition is based on the relation between entropies of the higher and lower hierarchical level.

In this paper, we make a contribution to the establishment of dynamical hierarchies by proposing a design principle for man–made systems which exhibit robust adaptation to environmental variation. In our investigation, the distinction is made between environmental variations whose effects can

 ISBN: 978-1-61208-109-0 

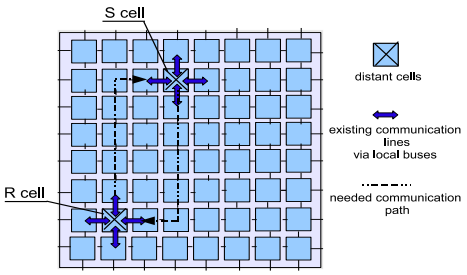



Figure 1: Schematic view of the system architecture

be seen on whole populations and those with effects on particular individuals. The former we term environmental changes and they are assumed to be tackled by the long term, (co)evolutionary processes; the latter we further refer to as environmental fluctuations and presume them tackled by the short term adaptive processes. Our work, as further described, considers the short term adaptive processes which take place in response to environmental fluctuations.

## III. Modelling Adaptive Processes for the Preservation of Homeostasis

Our approach has a starting point in the framework presented in [5]. It is explained in greater detail in [1], [2]. At this place, the main characteristics of the introduced model are repeated for the ease of understanding the work presented further in the paper. The system architecture under investigation is schematically represented in Figure 1. Such architecture can be related to cellular automata (CA) formalism [21] where each cell communicates only with the nearest neighbours in its Von Neumann neighbourhood. Its behaviour is performed and monitored over the discrete time steps (ticks). The system is assumed to perform some functionality which is the result of the functionality performed by each of its cells. Therefore, if the functionality of one of the system cells deviates from the desired, the functionality of the system as a whole is also affected.

Further, each cell is assumed to possess a sensor through which it can sense fluctuations in its local environment. Two types of identifiers are used for the cell identification: one referring to the physical position of the cell within the system, *physical ID*, and another to its functional relatedness to other cells, *encoding ID*. Functional relatedness defines from which cell the functional input is received and to which cell the functional output is sent. More precisely, the cell with the *encoding ID* $n$ is functionally related to the cells with the *encoding IDs* $n-1$ and $n+1$. Finally, the system configuration is given as a sequence of numbers where the position within the sequence corresponds to the cell's *physical ID* and the actual value represents its *encoding ID*.

The cell behaviour is defined by the finite state machine,

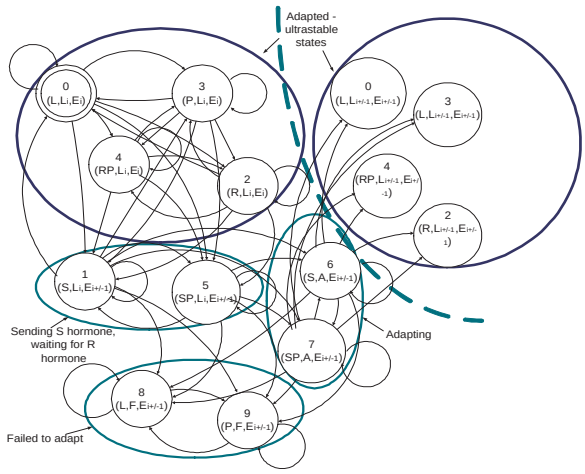



Figure 2: The finite state machine diagram describing the cell behaviour [22]

see Figure 2. The cell state is represented by a 3-tuple (H,A,E): H referring to hormonal flows; A to the cell functionality and E to the value of some environmental parameter which describes the effect of the fluctuation in the cell's environment on the system. Possible values for each member of a 3-tuple are as follows:

- H: L - no hormone present, S - sending *S hormone*, R - sending *R hormone*, P - passing hormone not functionally related to the cell, SP - sending *S hormone* and passing functionally unrelated hormone, PR - sending *R hormone* and passing functionally unrelated hormone;
- A: L0, L1, L2, L3 or L4 - functionality adapted to E0, E1, E2, E3 and E4 respectively, A - adapting or F - failed to adapt;
- E: E0, E1, E2, E3 and E4 - five different values of environmental parameter under consideration.

Control variables for the state transitions are as follows: the change in the environmental parameter (EE), the cell's *S hormone* present (SM), the cell's *R hormone* present (RM), hormones from functionally unrelated cells present (HO), incoming S hormone recognised (RR), incoming R hormone recognised (FB), hormone(s) from functionally unrelated cell(s) present at inputs (HI). They are omitted from Figure 2 for the sake of clarity.

According to the state diagram in Figure 2, the cell adapted to its environment will be in one of the ultrastable states (states 0,2,3,4), while upon detecting environmental fluctuation, it will start sending the message on possible deviation in its functionality (states 1 and 5) and perform adaptation process (states 6 and 7) after it receives information on its functional deviation. This information is provided by its functionally related cell, see Figure 1. If it fails to adapt, it will remain in one of the states 8 and 9. However, if it successfully achieves adaptation, it will move to one of the ultrastable states of another set pertaining to the new

 ISBN: 978-1-61208-109-0 

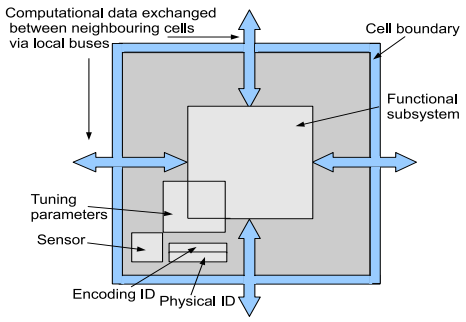



Figure 3: Schematic view of the cell architecture corresponding to the model of chemoton [23]; equivalent subsystems in chemoton model are given in parentheses

value of the environmental parameter. These states are shown to the right from the dashed line in Figure 2. Such behaviour is in accordance with [5].

We have further addressed the organisation of living systems for its autonomous and self-referring character. The cell architecture is related to the model of minimal living system provided by the chemoton theory [23]. Therefore, it is assumed that the cell consists of three subsystems, see Figure 3. The subsystem of tuning parameters is sensitive to environmental variations. It steers the operation and determines the performance of the cell's functional subsystem. The third subsystem - membrane, is rather straightforward to explain when the cell boundaries are considered. Additionally, the flow of nutrients and metabolic waste through the chemoton membrane corresponds to the flow of computational and control (hormonal) data via local communication lines.

The behaviour of the system can be described by the finite state machine diagram shown in Figure 4. The system can be in one of the three possible states:

- state 0: the system is adapted to its environment - none of the system cells detects environmental fluctuation, neither exhibits functional deviation
- state 1: the system performs adaptation process - at least one of the system cells detects variation in its environment and undergoes adaptation process
- state 2: the system fails to adapt - either $1/3$ of the system cells failed to achieve adaptation to the detected environmental fluctuation or $2/3$ of the cells detect environmental fluctuation

Such view of the system states can be related to the *levels of living* and *life criteria* as given in [24]. There, the state 0 would correspond to *the state of living*, the state 1 to *the state of being capable of living* and the state 2 to *the state of being dead*. As in Figure 4, once the system reaches this state, it remains in it i.e., it can not recover its functionality any more. On the other hand, state 1 still allows the system to achieve adaptation and eliminate functional deviation.

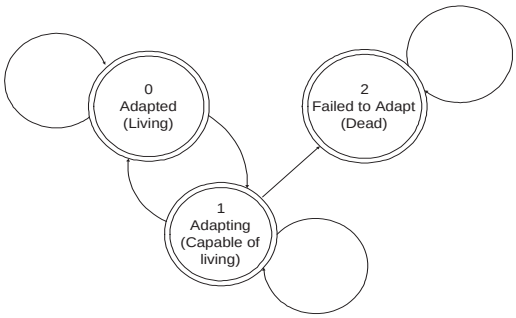



Figure 4: The finite state machine diagram describing the system behaviour

Based on such system behaviour, *we define system robustness to environmental fluctuations as its ability to avoid the state of failure* (the state 2 in Figure 4). Further, we introduce the degree of robustness as the probability to avoid the state of failure: *the greater the probability the system escapes from reaching the state 2, the greater the level of robustness it exhibits towards environmental fluctuations.*

### A. Adaptation Process and the Role of Hormonal Loops

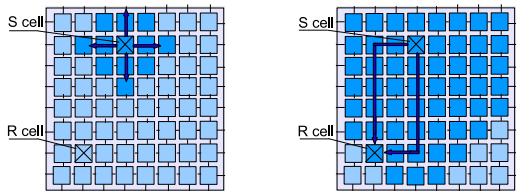
Adaptation process performed by the cell was described and demonstrated in simulation in [1]. In [2] it was reconsidered from the perspective of hormonal flows where hormonal loops were recognised to initiate and sustain adaptation process until adaptation is achieved. Hormonal loops are formed between functionally related cells through the following sequence of events:

1) the cell (*S cell*) senses environmental fluctuation
2) *S cell* begins secreting *S hormone* as a response to the sensed fluctuation, see Figure 5a
3) *S hormone* reaches the functionally related cell (*R cell*)
4) *R cell* recognises *S hormone*, see Figure 5b
5) *R cell* begins secreting *R hormone* which carries the information on the functional deviation exhibited by the related *S cell*, see Figure 5c
6) *R hormone* reaches *S cell*, see Figure 5d

Once the hormonal loop is closed, *S cell* begins adaptation process. This process can be performed provided two hormones are present in the cell: its *S hormone* and the *R hormone* from its functionally related cell. If any of these hormones is cleared from the system before adaptation is achieved, the cell fails to adapt. However, if both hormones are present during adaptation process until no functional deviation is detected by the related *R cell*, S cell achieves adaptation to environmental fluctuation which triggered the whole process.

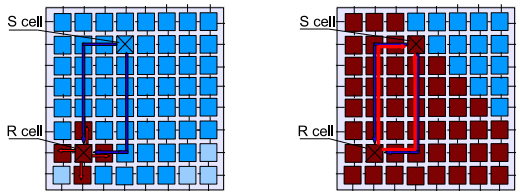
Figure 6 shows the loops for the configuration used in simulations in [2]. There, the value of the environmental parameter was changed for the cells with *encoding IDs* 52, 23 and 32. The loops which were formed between these cells and their corresponding functionally related cells can

 ISBN: 978-1-61208-109-0 




(a) S cell senses environmental fluctuation and begins secreting S hormone

(b) R cell recognises S hormone as coming from the functionally related cell




(c) R cell begins secreting R hormone

(d) R hormone reaches functionally related S cell - hormonal loop is closed

Figure 5: A loop formation: the flows of hormones which form the loop

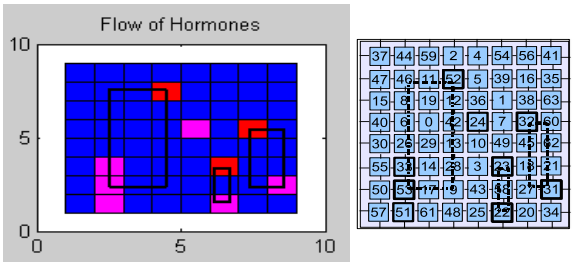
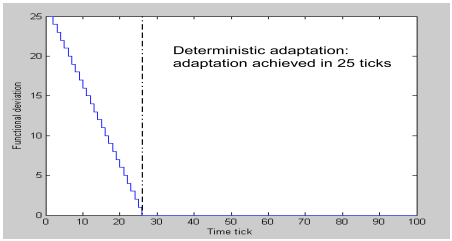



Figure 6: Hormonal loops as observed during one of the simulation runs for the system configuration to the right

be seen on the plot on the left side in Figure 6. This figure represents a plot of the system state pertaining to the *flow of hormones*, H, part of the state 3–tuple (H,A,E) during simulations. The amount of hormones flowing around the architecture during adaptation process is determined by the exponential decay rate parameter. This value can be made adaptive so that the presence of the hormone is ensured until adaptation is successfully achieved [1].

We denote a hormonal loop as $(S_i(n), R_i(n+1))$ if it is formed between *S cell* with the *encoding ID* $n$ and *R cell* with the *encoding ID* $n+1$. Index $i$ serves for the loop identification and corresponds to the order in which the loop was formed relative to other loops during simulation.

The progress of the adaptation process performed by the cell is modelled by a counter which is stochastically



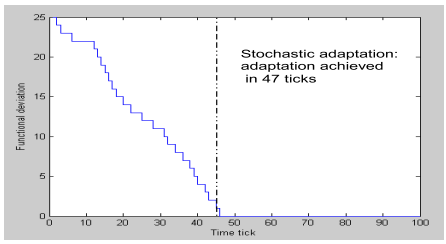




Figure 7: Model of the stochastic decrease in functional deviation during adaptation process

decremented each time tick during this process. The counter value reflects the functional deviation exhibited by the cell: upon the change in the environmental parameter value, this counter is assigned some value which represents maximum functional deviation exhibited by the cell; decrementing this value corresponds to the decrease in functional deviation as a result of the ongoing adaptation process. Figure 7 shows how this is performed for the case of maximum counter value 25 which was used in our simulations.

## IV. Enhancing the System Model – Dynamical Hierarchies Included

We pose the question if the adaptation process performed by the model introduced in Section III can be improved with respect to the exhibited robustness if the principles of hierarchical organisation are included. To investigate such possibility, the existing model need be enhanced so as to account for hierarchical organisation. However, for such an endeavour there are no straightforward solutions. In [25], preliminary considerations were presented concerning different ways in which hierarchies could be included. There, a number of arguments is provided in favour of dynamical against structural hierarchies. Primarily, for the environmental *dynamics* (fluctuation) we seek *dynamic* response of the system. Therefore, we look into the system dynamics initiated by environmental dynamics.

Several issues need be addressed. Firstly, higher level entities need be recognised: what entities arise as a result of the cell dynamics in response to environmental fluctuation? Secondly, the behaviour of these entities must be addressed: how do these entities behave, how do they interact? Does such behaviour bring in some novelty? If so, how can this

 ISBN: 978-1-61208-109-0 

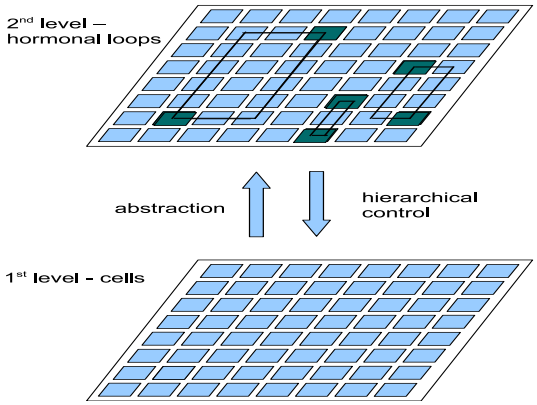



Figure 8: Schematic view of the hormonal loops observed as entities at the second hierarchical level

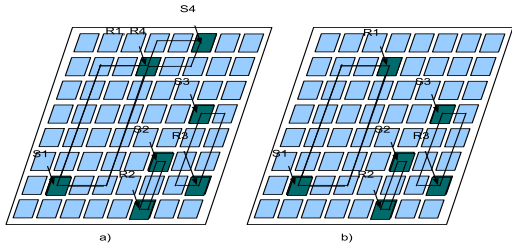



Figure 9: Hormonal loops: a) four loops out of which 2 are related i.e., loops $(S1, R1/R4)$ and $(S4, R1/R4)$ where $R1/R4$ stands for the cell which recognises incoming *S hormones* from both cells $S1$ and $S4$; b) three unrelated loops. The cells *encoding IDs* are omitted for the sake of clarity

novelty be used for the increase in the robustness of the performed adaptation process?

Further, when the description of such system is considered, different levels of description need be provided. This requires the granularity of each description level be chosen. A careful choice has to be made on what information from the lower level is to be discarded when describing the higher level dynamics. Finally, as our particular interest lies in adaptive processes, we address the issue of hierarchical control for such a process: what kind of hierarchical control could be imposed from a higher hierarchical level on the adapting cell at the lower level so that it accounts for the dynamics of higher level entities and improves robustness of the performed adaptation process? Figure 8 shows schematically relations between the two levels within the proposed model. Subsections IV-A and IV-B further explain the abstraction and the hierarchical control denoted by the upward and downward pointed arrows in this figure.

### A. Describing Behaviour from a Higher Level

The system responses to environmental dynamics through a sequence of hormonal secretions as described in section III-A. Therefore, we seek the candidates for the higher level entities among these hormonal flows. Hormonal flows which arise in the system upon the cell sensing the fluctuation in its environment are given in Figure 5 and introduced in section III-A. At first, one cell which secretes *S hormone* does not possess enough information to perform any meaningful action within adaptive process (Figure 5a). The information on the functional deviation exhibited by the *S cell* is available only after its functionally related *R cell* has recognised the incoming *S hormone* and began secreting *R hormone* (Figures 5b and 5c). This information can be provided by the related *R cell* due to the functional relatedness between the cells. However, although available within the system, it has no effect as long as it does not reach the cell affected by environmental variation. It is only at this point that the cell can begin its adaptive process. Hormonal loop is closed between the two functionally related cells (Figure 5d) one of which exhibits deviation in its functionality due to the environmental fluctuation, while the other provides information on this deviation which is necessary for the adaptive process to be performed. Hormonal loops can not be observed at the cell level but it takes a view from a higher level to observe them.

Next issue to address is the hormonal loop dynamics. The new level of description is needed which describes the behaviour related to the level of hormonal loops. This description level concerns the dynamics of the higher level entities – hormonal loops. Relevant information for the hormonal loop behaviour is contained in hormonal flows which make this loop as well as the status of the adaptation process. Hormonal flows define the loop existence: dependent on the flows of hormones, the loop will come into existence but also disappear. Moreover, the achievement of adaptation and failing to do so will affect the loop existence and behaviour.

When hormonal flows form the loop, the information on functional deviation is provided and the adaptation process may be performed. Further question may be posed if it is possible to enhance this information so that the adaptation process is improved. The information on the functional deviation is provided by the functionally related *R cell*. Can situations occur when this cell possesses more information on functional deviation performed by the related *S cell* than it otherwise would when based only on the functional relatedness between these two cells?

In a broader picture, because of the way in which the system configuration is implemented, one cell is functionally related to two other cells. Therefore, the *R cell* within the loop will be functionally related to one more cell beside the *S cell* within the loop. Dependent on the environmental dynamics, this other functionally related cell may also be affected by environmental fluctuation causing it to start

 ISBN: 978-1-61208-109-0 

secreting *S hormone* upon which its related *R cell* would secrete hormone with the information on its functional deviation as well. In that way, one *R cell* will be part of two hormonal loops, as schematically shown in figure 9. Because the cell functionality is dependent on the functionality of its predecessors as well as its successors in the configuration string, the *R cell shared* between two loops may provide each of the related *S cells* with more information on their functional deviation. Therefore, we introduce interaction between the loops based on the functional relatedness of their pertaining cells: *two hormonal loops will interact if their S cells share the information provided by the R cell which is part of each of the two loops simultaneously*. Such loops are said to be related. Figure 9 schematically shows the case where related hormonal loops are identified (a) and when no related loops exist (b).

The finite state machine diagram which describes the loop behaviour is shown in Figure 10. When no environmental fluctuation is detected, no hormonal flows are present and the loop is in the state 0. Such case corresponds to the cases when the cell which may initiate creation of hormonal loop, i.e., the *S cell* of the loop, is in the state 0, see Figure 2. The loop remains in the state 0 also when this cell is passing a hormone from another functionally unrelated cell (state 3 in Figure 2), when secreting *S hormone* without having recognised *R hormone* (state 1), when recognising the incoming *S hormone* (state 2) or corresponding to both first and either of the two mentioned cases (states 5 and 4 respectively). The loop comes into existence when the cell which is already secreting *S hormone* detects and recognises the incoming *R hormone*. Then, the loop moves to either of the states 1 or 2. This transition corresponds to the cell state transitions to states 6 and 7 in Figure 2, the exact transition being dependent on the presence of some hormone from a functionally unrelated cell: when such hormone is present, it corresponds to the cell state 7 and when not, to the state 6. As said, the loop also moves to one of the two states but which one in particular is determined by somewhat different criteria from the presence of the hormone from a functionally unrelated cell. If the *R cell* pertaining to the loop is already part of another loop, the loop will move to the state 2. Otherwise, it moves to state 1. In this state, the *S cell* is the only cell to which the *R cell* provides the information on functional deviation. If any of the hormones forming the loop is cleared from the system before adaptation is achieved, the pertaining *S cell* fails to adapt so the whole loop fails as well and it moves to the state 3 in Figure 10. This state corresponds to the cell states 8 and 9 in Figure 2. Again, which of the two in particular is dependent on the presence of the hormone from the functionally unrelated cell: it moves to the state 8 when no such hormone is present and to the state 9 when it is present. The loop may also move to the state 3 from the state 0. Such situation corresponds to the case when the cell begins secreting *S hormone* upon

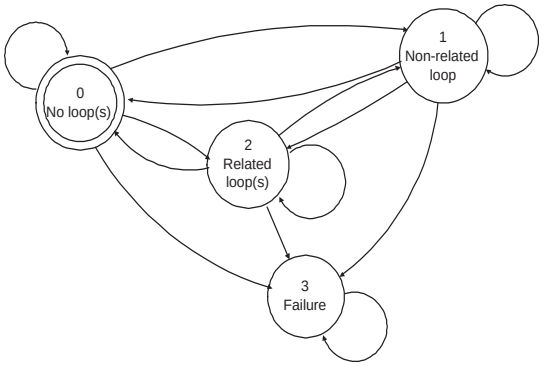



Figure 10: The finite state machine diagram describing the hormonal loop behaviour

the detection of environmental fluctuation but it does not begin to adapt because one of the two hormones needed to sustain adaptation process is cleared from the system before the loop is closed. It can be noticed that the behaviour at the loop level can not be fully described without the knowledge on the existence of the related hormonal loops. This concerns transitions between states 1 and 2. As the existence of the related hormonal loops can be observed only at the hormonal loops level, it may be said that the hormonal loops relatedness arises at this higher hierarchical level.

Comparing Figures 2 and 10, it can be noticed that the description of the behaviour at the higher, hormonal loop level (Figure 10), contains much less detail than the description of the behaviour at the lower, cell level (Figure 2). This is in accordance with [15] where the loss of detail at different description levels is explained from the physical point of view and related to the loss of information at different description levels respectively [20]. Such behaviour brings in new means of communication within the layer, realised through the interaction between hormonal loops, but, as explained further, also between the layers through the hierarchical control of adaptation process which accounts for the hormonal loops dynamics.

### *B. Hierarchical Control of Adaptation Process*

In [15], the role of hierarchical control within hierarchically organised systems is discussed. Here, the possibility of using the information generated by the higher level dynamics is examined with the aim of improving the adaptation process within the system. As mentioned, adaptation process performed by the cell is dependent on the existence of the hormonal loop to which the cell belongs as its *S cell*. Moreover, it was explained how the existence of related loops can provide more information on the functional deviation exhibited by the cell. Therefore, in cases when related loops exist for the loop to which the adapting cell belongs, this cell stands better chances to adapt due to the availability of

 ISBN: 978-1-61208-109-0 

a larger amount of information on its functional deviation. So, we suggest the following scheme for performing the adaptation process which accounts for the hormonal loops dynamics:

- If adaptation process is performed within the loop which does not detect related loop, the counter modelling this process is stochastically decremented by 1. Such scheme is depicted in Figure 7.
- If adaptation process is performed within the loop for which a related loop exists, the counter is decremented by 3 thereby modelling the case when more information is available for the achievement of adaptation.

In short, the more information available from the higher hierarchical level, the larger the adaptation increments are i.e., the larger the decrements in functionality deviation. For the example given in Figure 9 a), this means that the cells $S1$ and $S4$ would have the functional deviation value decremented by 3 during adaptation process while the cells $S2$ and $S3$ would have this value decremented by 1. This is because the cells $S1$ and $S4$ are part of the loops which share the *R cell* $R1/R4$. The novel feature brought into the system model would then be the hierarchical control determined by the second level dynamics which accounts for the interaction and relatedness between the second level entities – hormonal loops.

## V. Experimental Setup

The system is exposed to environmental fluctuations and its behaviour observed until its failure. To investigate how useful the information on the higher level dynamics can be for the hierarchical control of adaptation process as well as its effect on the system robustness, two types of simulation runs are performed:

- type 1: hierarchical control of adaptation process does not account for the interactions between the higher level entities – hormonal loops
- type 2: hierarchical control takes into account interactions between hormonal loops

The time ticks of the system failure are compared for the two types of run. If the information on the interactions between the higher level entities helps the system improve its robustness towards environmental fluctuations, the system will fail at a later point in time i.e., a later time tick when this information is accounted for than when it is not.

Exponential decay rate parameter is kept constant thereby allowing the cells to fail to adapt in some cases due to the insufficient amount of hormones. The value of this parameter was set to 0.9 based on the consideration of optimal values, see [1].

We define that the system fails to adapt if either of the following two conditions is fulfilled:

- one third of the system cells fails to adapt; for our system of 64 cells (a grid of $8X8$ cells), this value equals approximately 21
- two thirds of the system cells are sensing the fluctuation in the environmental parameter; for the system under investigation this value equals approximately 43

The numbers of the system cells which failed to adapt i.e., which are affected by the environmental fluctuation are chosen arbitrarily.

The effect of environmental fluctuations on the system is modelled by a change in the value of the parameter E, see section III. So, the system is being driven to *the state of being dead*, see Figure 4, by imposing more and more changes in the value of this parameter. The rate of change of the parameter value reflects the harshness of the environmental dynamics. It is expressed as a parameter $R_C$ which stands for the percentage of occurrences of environmental variations and is set to $5\%$ for the pertaining simulations. At each time tick, according to the $R_C$ value, the environmental parameter $E$ is changed with some probability $p(n)$. This probability models the stochastic nature of the environment. In the beginning, it is as low as 0.001. As the time proceeds, it is increased also with certain probability $p_{st}$ by a small amount i.e., 0.0005. Therefore, for each cell $i$, the probability of having the local environment changed may be expressed as:

$$p_i(n) = p_i(n-1) + incr_i(p_{st}), p_i(0) = 1e-3 \quad (1)$$

where $incr_i$ represents the increments added to the preceding time tick probability with some probability $p_{st}$:

$$\begin{aligned} incr_i(p_{st}) &= 5e-4, p_{st} \geq th \\ &= 0, p_{st} < th \end{aligned}$$

where $th$ is a threshold value. In this fashion, the total number of cells affected by the fluctuation in their local environment increases with time. In a real case, where the system tissue is rather of a continuous nature, it would be more realistic to speak of the gradient of environmental parameter over the system architecture. So, to account for this, we assign a greater probability of having the environmental parameter changed to those cells which are in the neighbourhood of the cells already affected by some environmental fluctuation. For these cells we set the threshold value to be higher than otherwise: 0.8 and 0.5 respectively.

In order to achieve statistically relevant results, this procedure is repeated for $10000$ configurations. The value of $10000$ was chosen based on the findings for the experiments with the same architecture within our previous work [1].

## VI. Results

The simulation results are summarised in table I. It shows averaged values of the time ticks when the system failed. The first row refers to the cases when the hierarchical control did not account for the interactions between second level entities during adaptation process, while the second row shows the result for the cases when it did. It can be noticed

 ISBN: 978-1-61208-109-0 

Table I: Numeric data on the system failure for two types of simulation runs averaged over 10000 configurations, $R_C = 5\%$

| Type of the run | Average time tick when the system failed | Avrg # of related loops for equal ticks of fail. | Avrg # of related loops for different ticks of fail. |
|---|---|---|---|
| No loops interactions included in control | **6.13e2** | – | – |
| Loops interactions included in control | **6.46e2** | 2.13e-1 | 1.58 |

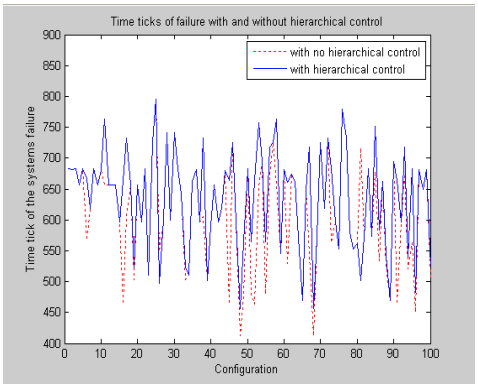



Figure 11: Time ticks of system failures for configurations used in simulations

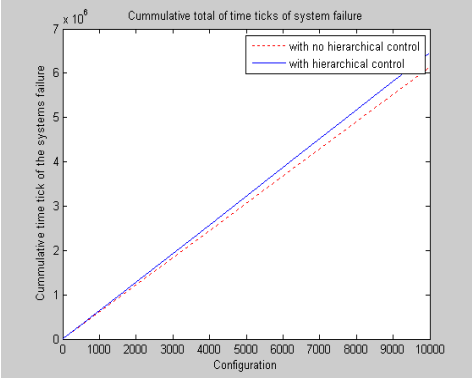



Figure 12: The cummulative time ticks of the system failure for configurations used in simulations

that the system fails at a later point in time i.e., a higher time tick when the hierarchical control is enhanced with the information generated by the interactions between the higher level entities. The results show that the time tick of the system failure is equal or higher when the interactions between the higher level entities are accounted for than when they are not. *This finding stands in support to the claim that hierarchies i.e., dynamical hierarchies can improve robustness in the assumed man–made system based on a grid-of-cells architecture.*

Figure 11 shows the values of the time ticks of failure

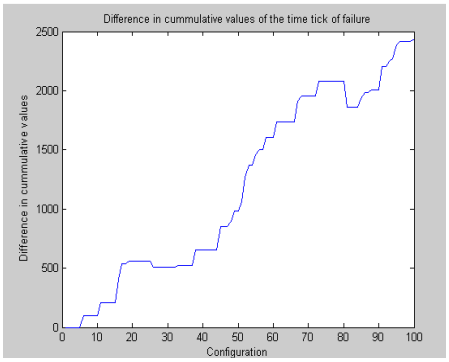



Figure 13: The difference in the cummulative time ticks of failure for the two types of adaptation processes - detail

for the first 100 configurations examined under simulations. It can be noticed that the tick of failure is equal or higher when the interactions between the higher level entities are accounted for than when not. Only 100 values are shown for the sake of clarity. However, the same trend holds for all configurations used in simulations. To visually support such observation, we present the accumulated value of the tick of failure for the two types of runs, see Figure 12. There, the graph corresponding to the case when the dynamical hierarchical control accounts for the interactions between the hormonal loops has a steeper slope throughout. In more detail, Figure 13 shows the difference between these cummulative values for the first 100 configurations.

## VII. Conclusion and Future Work

We have addressed the challenge of increasing robustness of a man–made system to environmental fluctuations. The assumed system was modular, its architecture based on CA. One possible way in which this could be achieved has been presented on previous pages. Moreover, the simulation results have shown that application of the principles of dynamical hierarchies can improve system robustness to environmental fluctuations. The improvement is due to the hierarchical control of adaptation process which accounts for the higher level dynamics while adaptation process is performed by entities at the lower level. At the same time, suggested solution has opened a number of questions which remain for further work to answer. One of the questions is related to the dynamics of environmental fluctuations. The results presented in the paper accounted for only one value of $R_C$. It would be interesting to look into the system behaviour under different environmental dynamics modelled by varying parameter $R_C$. Further challenge would be also to extend the investigation to even higher levels seeking general recommendations for building up a higher level dynamics.

On the other hand, this work has addressed the topic of dynamical hierarchies which has raised a great interest and inspiring discussions within the ALife research community

 ISBN: 978-1-61208-109-0 

over the recent years. We believe that the model presented in this paper might be the way towards bringing closer the theoretical research into dynamical hierarchies and engineering applications for man–made systems when needed to operate in dynamic environments. In particular, we have in mind applications where hardware and electronics systems need to operate not only in dynamic but also extreme, harsh environments.

Before we engage into addressing open challenges, the system model needs to be put within a firmer theoretical framework with dynamical hierarchies included. However, debates are still being held on this topic so such an endeavour is not a straightforward thing to do. Rather, to begin with, we have chosen one of the proposed approaches which is based on information theory [20]. Our deep belief is that in succeeding to do so, a firm theoretical basis for our research would be established opening new possibilities for further development of ideas.

## ACKNOWLEDGMENT

D.L. would like to thank Chrisantha Fernando for inspiring talks during ECAL 2009.

## REFERENCES

[1] D. Laketic, G. Tufte, and P. Haddow, "Stochastic adaptation to environmental changes supported by endocrine system principles," in *NASA/ESA Adaptive Hardware and Systems, AHS 2009*. IEEE Computer Society, 2009, pp. 215–222.

[2] D. Laketic and G. Tufte, "Adaptation in tissue sustained by hormonal loops," in *10th European Conference on Artificial Life*, 2009.

[3] T. Lenaerts, D. Gross, and R. Watson, "Introduction to the workshop wdh2002," in *Proceedings of the Alife VIII workshop On the modeling of dynamical hierarchies*, D. Gross and T. Lenaerts, Eds. Sydney: University of New South Wales, 2002, pp. 37–44.

[4] A. Guyton and J. Hall, *Textbook of Medical Physiology*. Elsevier Saunders, 2005.

[5] W. R. Ashby, *Design for a Brain, the origin of adaptive behaviour*. Chapman & Hall Ltd., 1960.

[6] J. Lovelock, *Gaia - A new look at life on Earth*. Oxford: Oxford University Press, 1979.

[7] M. Neal and J. Timmis, "Timidity: A useful emotional mechanism for robot control?" in *Informatica – special issue on perception and emotion based control*. The Slovene Society Informatika, Ljubljana, Slovenia, June 2003, pp. 197–204.

[8] W. M. Shen, B. Salemi, and P. Will, "Hormone-inspired adaptive communication and distributed control for conro self-reconfigurable robots," in *IEEE Transactions on Robotics and Automation*, October 2002, pp. 700–712.

[9] R. Moioli, P. Vargas, F. Von Zuben, and P. Husbands, "Towards the evolution of an artificial homeostatic system," in *IEEE Congress on Evolutionary Computation 2008, Proceedings*, June 2008, pp. 4023–4030.

[10] K. Suzuki and T. Ikegami, "Homeodynamics in the game of life," in *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*. MIT Press, Cambridge, MA, 2008, pp. 600–607.

[11] M. Asada, J. C. T. Hallam, J.-A. Meyer, and J. Tani, Eds., *Extended Homeostatic Adaptation: Improving the Link between Internal and Behavioural Stability*, ser. Lecture Notes in Computer Science. Springer, 2008.

[12] A. J. Greensted and A. M. Tyrrell, "Implementation results for a fault-tolerant multicellular architecture inspired by endocrine communication," in *NASA/DoD Conference on Evolvable Hardware, Proceedings*, June-July 2005.

[13] H. A. Simon, "The organisation of complex systems," in *Hierarchy Theory*, H. H. Pattee, Ed. George Braziller Inc., 1973, pp. 1–27.

[14] C. Grobstein, "Hierarchical order and neogenesis," in *Hierarchy Theory*, H. H. Pattee, Ed. George Braziller Inc., 1973, pp. 29–47.

[15] H. H. Pattee, "The physical basis and origin of hierarchical control," in *Hierarchy Theory*, H. H. Pattee, Ed. George Braziller Inc., 1973, pp. 71–108.

[16] D. Gross and T. Lenaerts, "Towards a definition of dynamical hierarchies," in *Artificial Life VIII Workshop Proceedings*, 2002.

[17] S. Rasmussen, N. Baas, B. Mayer, M. Nilsson, and M. Olesen, "Ansatz for dynamical hierarchies," in *Artificial Life*. Massachusetts Institute of Technology, 2001.

[18] D. Gross and B. McMullin, "Is it the right ansatz?" in *Artificial Life*. Massachusetts Institute of Technology, 2001, pp. 355–365.

[19] S. Rasmussen and N. Baas, "A defense of the ansatz for dynamical hierarchies," in *Artificial Life*. Massachusetts Institute of Technology, 2002, pp. 367–373.

[20] S. McGregor and C. Fernando, "Levels of description: A novel approach to dynamical hierarchies," in *Journal of Artificial Life (Species Issue on Dynamical Hierarchies)*, 2005, pp. 459–472.

[21] E. F. Codd, *Cellular Automata*, ser. Association for computing machinery, Inc. Monograph series. Academic Press, New York, 1968.

[22] D. Laketic, "Homeostatic adaptation to environmental fluctuations sustained by hormonal flows," Norwegian University of Science and Technology, Tech. Rep., 2010, in print.

[23] T. Ganti, *Chemoton Theory Vol.1., Theory of Living Systems*. Kluwer Academic/Plenum Publishers, 2003.

[24] ——, *Chemoton Theory Vol.2., Theory of Living Systems*. Kluwer Academic/Plenum Publishers, 2003.

[25] D. Laketic and G. Tufte, "Achieving robustness in adaptive systems – can hierarchies help?" in *ICAART 2010, 2nd International Conference on Agents and Artificial Intelligence, Proceedings*, 2010, pp. 422–427.

 ISBN: 978-1-61208-109-0 

# A QoI-aware Framework for Adaptive Monitoring

Bao Le Duc*, Philippe Collet‡, Jacques Malenfant† and Nicolas Rivierre*
*Orange Labs, Issy les Moulineaux, France
Email: {bao.leduc, nicolas.rivierre}@orange-ftgroup.com
‡Université de Nice Sophia Antipolis, CNRS, UMR 6070 I3S, Sophia Antipolis, France
Email: Philippe.Collet@unice.fr
†Université Pierre et Marie Curie-Paris 6, CNRS, UMR 7606 LIP6, Paris, France
Email: Jacques.Malenfant@lip6.fr

***Abstract*—Monitoring application services becomes more and more a transverse key activity in information systems. Beyond traditional system administration and load control, new activities such as autonomic management and decision making systems raise the stakes over monitoring requirements. In this paper, we present ADAMO, an adaptive monitoring framework that tackles different quality of information (QoI)-aware data queries over dynamic data streams and transform them into probe configuration settings under resource constraints. The framework relies on a constraint-solving approach as well as on a component-based approach in order to provide static and dynamic mechanisms with flexible data access for multiple clients with different QoI needs, as well as generation and configuration of QoS and QoI handling components. The monitoring framework also adapts to resource constraints.**

***Keywords*-Monitoring, Adaptive systems, Quality of information, Component framework**

## I. INTRODUCTION

As distributed and pervasive systems are now deployed everywhere with 24/7 availability constraints, monitoring becomes more and more a transverse key activity in enterprise computing. Beyond traditional system administration and load control, new activities increasingly require automated management of the systems, raising the stakes over monitoring requirements. Specific tasks such as scheduling, resource allocation and problem diagnosis make their decisions upon the online and continuous monitoring of the services, systems and infrastructures. Besides, autonomic management and decision making systems are now organized around *Service Level Agreements* referring to some *Quality of Service* (QoS) criteria. As large QoS variations are easily observable by clients when calling distant applications and services, there is also a large variation in the monitoring requirements, in terms of the types of monitoring data to be acquired, their lifespan, precision and granularity. This is generally referred as *Quality of Information* (QoI), i.e., an expression of the properties required from the monitored QoS data [1].

Moreover, deployment contexts have evolved in size and complexity, from intra-enterprise Service-Oriented Architectures (SOA) principles with low-latency network to large-scale inter-enterprise infrastructures with high latency, and finally to pervasive systems with dynamic contexts. Monitoring a distributed system involves extracting information among the deployed processes and their interactions, collecting it efficiently and making them available to the interested users in an appropriate format. The distributed context makes the monitoring activity inherently more complex than the more traditional centralized one, as it forces to handle several control flows, communication delays between nodes, nondeterministic event ordering and an extensive behavioral alteration on the observed system [2].

These challenges are hardly addressed by current monitoring systems. In a SOA context, prior works show that behavioral and basic QoS constraints can be expressed and monitored at runtime [3], [4], but with no QoI or only some implicit ones like statistics on QoS [5]. A monitoring system must currently provide several information flows to multiple clients, with different QoI requests, everything being dynamically reconfigurable. Finally, the monitoring system, being constantly operational, is itself subject to constraints on the resources it consumes to provide its services. Consequently, designing and deploying monitoring systems that are well-adapted to such requirements now become a complex and tedious activity for software architects and system administrators. Automation of this process is clearly needed. Recent works focus on QoI and adaptive monitoring for context-aware computing, data stream processing or transactional systems [6], [7], [8], but no monitoring system is currently adapted to all (changing) requirements together.

In this paper, we present ADAMO, an adaptive monitoring framework that tackles different QoI-aware data queries over dynamic data streams, transform them into probe configurations settings under resource constraints. This process relies on a constraint-solving approach. The framework also factors out the common structure and behavior of monitoring systems so that they can be reusable and extensible. To do so, it leverages component-based techniques so that a common base architecture is provided as an assembly of interacting components. Different parts of the architecture are then configurable, or can be partly generated from high-level descriptions of the monitoring requirements. ADAMO thus aims at providing solutions for i) flexible access to dynamic

 ISBN: 978-1-61208-109-0 

data streams for multiple clients with different QoI needs, ii) capability to take into account QoI constraints to generate and configure appropriate elements in the monitoring system, iii) making the monitoring system adaptable to resource constraints, and iv) ability to manage data queries in a static or incremental way. The rest of the paper is organized as follows. Section 2 motivates our work. The underlying QoI model and the base capabilities of the ADAMO framework are described in Section 3. Section 4 presents the ADAMO framework through its architecture and some illustration of its usage, as work in progress. Section 5 concludes this paper and discusses future work.

## II. Motivation

This section motivates our work by introducing a running example and surveying related work on adaptive monitoring.

### A. Motivating example

As a running example throughout the paper, we introduce a flood management system (inspired from the French ANR SemEUsE research project[1]). Such systems, known as $C^3$ (Control, Command and Communication), are mediators between commanders and their teams on the field. In flood management, organizing optimally rescue teams, transportation (boats to take people away from the dangerous zone), aerial means (helicopters) and medical teams require a lot of information, much of which coming from automatic sensors:

- GPS devices put on mobiles (boats, helicopters, personnel) sending positioning but also other data (fuel level, unused transportation capacity, ...) at some frequency;
- field sensors, measuring environmental data like water levels and their degree of variation, humidity indexes, rain levels, temperature, wind, etc.

They typically use data connections over GSM to transmit data at a frequency that can be set by instructions sent to them as messages. GSM networks tend to be overloaded in crisis situations, so the bandwidth is a scarce resource to be optimally used and bounded by some limit (e.g., 10% of the total bandwidth). When building situation reports, upon which commanders will decide, for example, which helicopter or which rescue team to send towards an emergency, it is crucial that the information presented be coherent, *i.e.,* illustrative of a coherent situation within some time interval, and not to old, *i.e.,* the age of the data does not pass some limit. As not all the data have the same importance, these parameters and the frequencies of their transmission they imply must be configured accordingly. For example, commanders may require a situation for helicopters (positions, remaining autonomy) coherent within 30 seconds and not older than 2 minutes, while for rescue teams these can be loosen to 2 minutes and 5 minutes respectively and for transportation teams, down to 5 and 10 minutes.

[1]http://www.semeuse.org

Moreover, as the authority structure is typically hierarchical, different commanding officers may require data with different QoI, depending upon their rank or their relation to the monitored entity. Higher rank officer have less stringent requirements, typically an order of magnitude less, when building aggregated global situation reports, while occasional requesters of a particular mean may be satisfied with less up-to-date data.

The overall goal of a monitoring framework as ADAMO is to build, configure and deploy the necessary components between the application and the sensors, and configure these so to match the required QoI while respecting resource and deployment constraints. If the problem appears to be overconstrained, utilities can be assigned to the different data so to guide the tradeoffs between them when computing their transmission frequencies (see Figure 1).

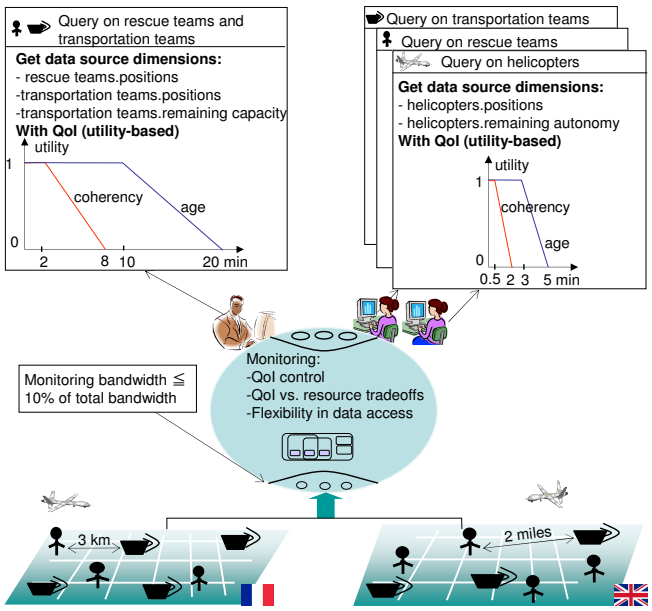



Figure 1. Query examples. In the left, a higher rank officer queries rescue and transportation team positions with less stringent QoI requirements; whereas in the right, lower rank officers query on resources they command with strict QoI requirements.

### B. On Adaptive Monitoring

Distributed monitoring is intrinsically a complex activity and current large scale architectures of distributed systems impose new requirements and strong constraints on monitoring. Consumers of the monitoring system are now **applications** and not only human users. Applications act as multiple clients, requesting for very different QoS data with specific QoI constraints on each of them. Moreover these applications make and change their queries dynamically, adding a new stringent requirement on the monitoring system. On the other side, **data sources** are also very varied and may be at different locations of the distributed system, thus impacting bandwidth consumption when data are larger

or transmission rate higher. Finally, system **administrators** needs to deploy the monitoring system under resource constraints, so that the overall resource consumption of the monitoring system itself is mastered during execution. Many research approaches have been proposed, to handle data collection from sources in context-aware computing, to add QoI capabilities on existing QoS monitoring systems, to provide adaptive monitoring infrastructures or also to build full adaptive systems.

To the best of our knowledge, no monitoring system is currently adapted to all these requirements. Nevertheless some systems provide really powerful solutions to one or several specific features of a monitoring system, from adaptive capabilities to QoI awareness or consumption regulation. We thus advocate a framework approach so that generic parts of a monitoring system can be more easily reused and extended and that well-adapted monitoring systems can be instantiated for specific needs. Consequently the framework must be able to deal with multiple clients needing flexible and dynamically reconfigurable access to dynamic data streams with different QoI needs, and to provide automatic configuration of all monitoring entities and data sources so that QoI and resource constraints are taken into account.

### C. Related Work

This section presents an overview of the research area on adaptive monitoring and QoI control in software systems.

Context-aware systems are concerned with QoI to perceive situations and adapt applications based on the recognized context. Quality of Context is well studied in [1], [9], [10] where many dimensions are proposed, including precision, freshness and consistency of the monitored data. These works, however, do not address the architecture of context-aware systems or the problem of maximizing QoI over a set of constraints.

Among the different works on context-aware management systems, Conan et al. [11] propose an architecture based on components (called context nodes) that are responsible to produce higher level context information from data gathered at lower architectural layers. The authors describe several patterns to compose the individual context nodes in order to implement the desired logic of a context-aware application. In [6], they show how to extend their approach to support Quality of Context (QoC) by using a specialized component to filter and evaluate QoC from collected information. Although they do not address the problem of maximizing QoI in overloaded situations their architecture is highly modular and extensible, and allows to introduce controlled tradeoffs between QoI requirements and resource consumption.

Poladian et al. [12], [13] focus on adaptive systems based on multiple concurrent applications running on local computing devices with limited memory, CPU and bandwidth. They propose an analytical model and an efficient algorithm to decide how to allocate scarce resources to applications, and how to set the quality parameters of each application to best satisfy user and supplier preferences. Their approach fits well into the framework proposed here to adjust the monitoring to the current conditions, given QoI objectives.

Data stream processing systems such as sensor networks or financial services are concerned with the problem of saving network or compute resource to deliver accurate information. Babcock et al. [14] propose a load shedding technique for continuous monitoring queries over data streams. The key idea is to carefully drop some tuples in order to reduce bandwidth and processing in overloaded situations. The authors formalize load shedding as an optimization problem with the goal of minimizing query inaccuracy within the limits imposed by resource constraints. Tatbul et al. [15] extend this approach for distributed stream processing systems. These works propose sophisticated algorithms and optimization techniques. However they do not address the design of the monitoring framework to implement them in a modular and flexible way, or focus on scalability issues in large-scale distributed stream processing systems [7].

Among the different works on predicting runtime malfunctions in software systems, Munawar et al. [8], [16] propose a new approach to monitor multi-tier transaction systems at a minimal level in normal condition and adaptively increase monitoring if a health problem is suspected. Their approach uses relationships between the monitored data in the form of regression models to determine normal operation and areas that need more monitoring in the event of anomalies. Their work fits well in presence of multiple metrics to dynamically adjust monitoring to the current condition but focuses on health prediction and doesn't consider QoI requirements such as age of the monitoring data.

## III. A QoI MODEL FOR ADAPTIVE MONITORING

This section presents ADAMO's QoI model, formalizing data sources, monitoring queries and system resources. The model leverages constraint solving to find appropriate frequencies to configure data sources according to clients needs and resource constraints.

### A. A Model for Adaptive Monitoring

Consumers send ADAMO QoI-aware monitoring queries and receive data streams as result. ADAMO hence addresses QoI by processing queries in such a way to automatically translate the requested QoI and resource constraints into data source configurations.

**Definition (Data source).** A data source $s$ is a triple $(\iota_s, \Phi_s, \Pi_s)$ where $\iota_s$ is a data source identifier, $\Phi_s = (\phi_{s,1}, ..., \phi_{s,n})$ is a data stream generated by $s$, and $\Pi_s$ is a set of constraints on data source properties.

In this model, monitored values are defined as independent data sources, even though some may report to the same physical entity. The data stream consists of sequences

 ISBN: 978-1-61208-109-0 

of data produced in temporal order by some measurement unit or probe. Each element $\phi_i$ contains a data value and a time-stamp representing when the value is generated to enforce QoI constraints. Constraints on data source properties express possible configuration settings, *e.g.,* interrogation mode (push/pull), sampling frequency... The latter are also used to regulate the monitoring and can be assigned a configuration value among the admissible ones for each data source through a configuration $C_s$ imposed at run-time. Sampling frequencies act as filters on the raw data stream to pick the values that will be transmitted to clients by the monitoring framework. It should be noted that $\pi_s$ denotes below the set of properties constrained by a data source.

*Example.* The data source for the remaining autonomy of helicopter 1 is $ha_1 = (h1\_aut,\ ((120, t_0), (118, t_1), ...),\ \{f_{ha_1} \in \{0.5, 1, 2\},\ msgSize_{ha_1} = 1\})$, where the remaining autonomy in the stream is expressed in minutes timestamped with $t_0$, $t_1$, ... (unspecified here), and where the frequency $f_{ha_1}$ and message size properties are constrained to be 0.5, 1 or 2 data per minute and exactly 1kb respectively. The set of data source properties $\pi_{ha_1} = \{f_{ha_1}, msgSize_{ha_1}\}$. □

**Definition (QoI-aware monitoring query).** A query $q$ is a couple $(\iota_q, \Pi_q)$ where $\iota_q = (\iota_{q,1}, ..., \iota_{q,n})$ is a set of sources from which the consumer wants to get data, and $\Pi_q$ is a set of QoI constraints imposed by the consumer on all of the data sources in $\iota_q$.

A query specifies the need of a consumer in the reception of tuples of data (required data sources) under the given QoI constraints. $\pi_q$ denotes below the set of QoI properties constrained by $\Pi_q$. Currently, ADAMO addresses two different QoI properties: age and coherency.

**Definition (Age and coherency constraints).** An age constraint imposes a maximal delay between the production of a data by a source and its reception by the consumer. A coherency constraint imposes a maximal delay between any pair of data for the requested tuple to be considered as valid.

*Example.* The aerial means officer needs helicopter 1 and 2 position and remaining autonomy not older than 2 minutes and a coherency of 30 seconds. The query is $((h1\_pos, h1\_aut, h2\_pos, h2\_aut), \{age \leq 2, coherency \leq 0.5\})$. The set $\pi_q$ of monitoring properties constrained by the query is $\{age, coherency\}$. □

**Definition (Resource).** A resource $r$ is a tuple $(\iota_r, \Pi_r, \upsilon_r, \oplus_r)$ where

- $\iota_r$ is a resource identifier,
- $\Pi_r$ is a list of data source properties impacting the consumption of the resource $r$,
- $\upsilon_r$ is a function of the properties $\Pi_r$ giving the consumption of the resource $r$ by a data source $s$ given the settings of its properties $\Pi_r$, and
- $\oplus_r$ is an aggregation function to combine the consumptions of data sources into an estimation of the global resource consumption of the monitoring system.

**Definition (Resource constraints).** Let $R$ be a set of resources used by the monitoring, $\mathcal{C}_R$ is a set of constraints put on these resources.

System resources used by the monitoring encompass bandwidth, CPU, memory... Each of the resources uses available data source properties expressing the consumption of that resource when delivering data to consumers to get the overall estimation of their consumption by the monitoring system in a given configuration of the data sources.

*Example.* Consider the case where the bandwidth used by the delivery of monitoring data must be kept under 10% of the total bandwidth of the network. The bandwidth resource is defined by $b = (bandwidth, \{f, msgSize\}, \upsilon_b, sum)$ where $\upsilon_b(f, msgSize) = f \times msgSize$ and $sum$ simply says that bandwidth consumptions of data sources are summed to get the overall bandwidth consumption of the monitoring. If the total bandwidth is $TB$, the constraint is $C_R = \{bandwidth \leq 0.1TB\}$. □

We denote $Q$ a set of monitoring queries and $S$ a set of data sources. $S_Q$ is the subset of $S$ used by $Q$. The principal challenge for adaptive monitoring is to find a data source configuration $C_{S_Q}$ satisfying a given set of queries $Q$ under the resource constraints $C_R$.

### *B. QoI-aware Control Capability*

The above model is generic and open to extend to new data sources, properties, resource and constraints. As QoI is concerned, ADAMO nevertheless considers age and coherency as primary properties. This section shows how the constraints on these are dealt with in the current implementation of ADAMO. The first lesson learned is that each kind of QoI requires a specific processing, hence extensibility of the platform with regards to QoI and how it is handled is mandatory. To put forward this extensibility requirement, we now introduce an approach to the model resolution in two contexts. First, we look at a resource unconstrained case, and then we add the resource constraints.

In the first context, the system is assumed to have sufficient resources in order to process all data queries. In this case, for any $s$, $C_{S_Q}$ is a configuration that satisfies highest QoI requirements among the set of queries $Q_s$ using $s$. In the second context, resources are constrained, computing $C_{S_Q}$ amounts to find a trade-off between QoI requirements and resource constraints. This trade-off problem varies upon usage contexts as well as how QoI impacts on consumers. For example, when the system has not enough resources, a simple approach is to reduce QoI equally for

 ISBN: 978-1-61208-109-0 

all consumers. Whereas in utility-based systems [12], [17], some requirements can have higher utility than others (*e.g.,* rescuing people versus rescuing animals). Consequently, utilities lead the monitoring to guarantee higher QoI for certain consumers at the expense of reducing it for the rest.

In both cases, frequency $f_s$ of any source $s$ is computed in order to achieve the QoI required by the set of queries $Q$ (message size could also be assigned, but here all of these constraints are equalities, so imposing a single value).

*QoI Enforcement in a resource unconstrained system*

When resource is not a concern for the monitoring, given a set of queries $Q$, the problem is to find an assignment for all the properties of each data source $s \in S_Q$ such that the constraints $\Pi_s$ and $\Pi_q$ are satisfied for all $s \in S_Q$ and all $q \in Q$. We model the problem as a *constraint satisfaction problem* (CSP). CSP is particularly well-adapted to ADAMO, as it provides a methodical approach to the problem, paving the way to extensions, such as integrating resource constraints (done next) but also to new types of constraints like cross-constraints among the different criteria and on other QoI when needed by the users. We now define such a CSP from data sources and query constraints.

The variables in the CSP are the data source and the QoI properties appearing in the data source and QoI constraints. Constraints $\Pi_s$ put on data sources are simply imposing restrictions on the domain of the configuration variables of the data source. They can be used as is in the CSP. Constraints $\Pi_q$ on the QoI need to be related to the configuration properties of data sources in order to enforce some values for their configuration. Under the hypothesis that the data sources cannot be synchronized in any way, one can see that any frequency of the data source large enough to produce data with a time interval between two values that exceeds neither the age nor the coherency constraints is admissible to configure the data source. This observation leads to the following formulation of the resource unconstrained data source configuration problem.

**Definition (CSP formulation, unconstrained case).** Let $Q$ be a set of monitoring queries and $S_Q$ the set of resources required by $Q$, the CSP formulation of the problem is:

1) The set of variables of the problem is
$$\bigcup_{q \in Q} \pi_q \cup \bigcup_{s \in S_Q} \pi_s$$
2) $\forall s \in S_Q$, the constraints $\Pi_s$ are added to the CSP.
3) $\forall q \in Q$, let $a_q \leq v \in \Pi_q$ be the age constraint of $q$, then the constraints $a_q \leq v$, and $\forall s \in S_q$, $f_s \geq 1/a_q$ are added to the CSP.
4) $\forall q \in Q$, let $c_q \leq v \in \Pi_q$ be the coherency constraint of $q$, then the constraints $c_q \leq v$, and $\forall s \in S_q$, $f_s \geq 1/c_q$ are added to the CSP.

The CSP obtained using the above definition may not have only one solution, as multiple frequencies for data source may match the desired age and coherency constraints of the queries. In this case, we choose the smallest frequencies in the sets of values satisfying all of the constraints.

*Example.* Consider ten data sources and three queries from the flood fighting scenario described above. Data sources are position and remaining autonomy for helicopters ($hp$, $ha$), position of rescue teams ($rp$), and position and remaining capacity for transportation teams ($tp$, $tc$). Each query ($q_1$, $q_2$, $q_3$) specifies the sources from which the consumer wants to get data and the QoI constraints on age ($a_{q_i}$) and coherency ($c_{q_i}$) imposed by the consumer on all of these data sources.

$$\begin{aligned}
hp_1 &= (h1_pos, (...), \{f_{hp_1} \in \{1,2,5\}, msgSize_{hp_1} = 1\}) \\
ha_1 &= (h1_aut, (...), \{f_{ha_1} \in \{1,2,5\}, msgSize_{ha_1} = 1\}) \\
hp_2 &= (h2_pos, (...), \{f_{hp_2} \in \{1,2,5\}, msgSize_{hp_2} = 1\}) \\
ha_2 &= (h2_aut, (...), \{f_{ha_2} \in \{1,2,5\}, msgSize_{ha_2} = 1\}) \\
rp_1 &= (r1_pos, (...), \{f_{rp_1} \in \{1/2,1,2\}, msgSize_{rp_1} = 1\}) \\
rp_2 &= (r2_pos, (...), \{f_{rp_2} \in \{1/2,1,2\}, msgSize_{rp_2} = 1\}) \\
tp_1 &= (t1_pos, (...), \{f_{tp_1} \in \{1/5,1/2,1\}, msgSize_{tp_1} = 1\}) \\
tc_1 &= (t1_cap, (...), \{f_{tc_1} \in \{1/5,1/2,1\}, msgSize_{tc_1} = 1\}) \\
tp_2 &= (t2_pos, (...), \{f_{tp_2} \in \{1/5,1/2,1\}, msgSize_{tp_1} = 1\}) \\
tc_2 &= (t2_cap, (...), \{f_{tc_2} \in \{1/5,1/2,1\}, msgSize_{tc_1} = 1\}) \\
q_1 &= (\{rp_1, rp_2, tp_1, tc_1, tp_2, tc_2\}, \{a_{q_1} \leq 10, c_{q_1} \leq 2\}) \\
q_2 &= (\{hp_1, ha_1, hp_2, ha_2\}, \{a_{q_2} \leq 2, c_{q_2} \leq 1/2\}) \\
q_3 &= (\{hp_1, ha_1, hp_2, ha_2, rp_1, rp_2\}, \{a_{q_3} \leq 2, c_{q_3} \leq 1/2\})
\end{aligned}$$

The set of constraints of the CSP includes all of the domain constraints of the ten data sources as well as the QoI property constraints of the three queries, to which are added the following constraints linking QoI to data source properties:

$$\begin{array}{rcl@{\quad}rcl}
q_1 : f_{rp_1} & \geq & 1/10 & f_{rp_1} & \geq & 1/2 \\
q_1 : f_{rp_2} & \geq & 1/10 & f_{rp_2} & \geq & 1/2 \\
q_1 : f_{tp_1} & \geq & 1/10 & f_{tp_1} & \geq & 1/2 \\
q_1 : f_{tc_1} & \geq & 1/10 & f_{tc_1} & \geq & 1/2 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
q_2 : f_{hp_1} & \geq & 1/2 & f_{hp_1} & \geq & 2 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\
q_3 : f_{hp_1} & \geq & 1/2 & f_{hp_1} & \geq & 2 \\
\ldots & \ldots & \ldots & \ldots & \ldots & \ldots
\end{array}$$

which simplifies to:

$$\begin{array}{rcl@{\quad}rcl}
f_{hp_1} & \geq & 2 & f_{ha_1} & \geq & 2 \\
f_{hp_2} & \geq & 2 & f_{ha_2} & \geq & 2 \\
f_{rp_1} & \geq & 2 & f_{rp_2} & \geq & 2 \\
f_{tp_1} & \geq & 1/2 & f_{tc_1} & \geq & 1/2 \\
f_{tp_2} & \geq & 1/2 & f_{tc_2} & \geq & 1/2
\end{array}$$

Taking the minimal frequencies satisfying these constraints, data sources of helicopters and rescue teams will have their frequencies set to 2 data per minute, while transportation teams will be set to 1 datum every 2 minutes. □

*QoI Enforcement in a resource constrained system*

Given a resource $r$ as defined in the section III-A, we now consider an amount $A$ of resource $r$ is allocated to the

monitoring. At first sight, we just need to add the following constraint (to simplify the notation, assume $\oplus_r$ is a sum) to the constraint system elaborated for the unconstrained case:

$$\sum_{s \in S_Q} v_r(\Pi_r|_s) \leq A$$

where $\Pi_r|_s$ are the properties that $r$ depends upon for the data source $s$. The problem is then to find a configuration $\mathcal{C}_{S_Q}$ that satisfies not only the age and coherency constraints, but also this resource constraint.

*Example.* For the case of the bandwidth constrained not to pass over 10% of the total bandwidth $TB$, we have $\Pi_r = \{f, msgSize\}$, $v(\Pi_r) = f \times msgSize$, and the aggregation function is a sum, hence the resource constraint becomes:

$$\sum_{s \in S_Q} f_s \times msgSize_s \leq 0.1 \times TB \quad (1)$$

□

However, the system being constrained in a new way, this can be considered to change the nature of QoI control problem. Indeed, as the resource constraint may impair the satisfaction of the age and coherency constraints of some queries, the user should be able to express preferences among its queries so to concentrate the resource on the most important queries and lower, if necessary, the requirements of the less important ones.

In order to allow the user to express his/her preferences over QoI properties, the query definition is extended with a set of utility functions $U_q$ that contains one utility function $\mu_{q,p}$ for each monitoring property $p \in \pi_q$. $\mu_{q,p}$ maps configurations $C_{\iota_q}$ of data sources used in $q$ to a utility value in $\mathbb{R}$. These utilities are combined to get the total utility of a configuration as follows:

$$U_Q(C_{S_Q}) = \sum_{q \in Q} \prod_{p \in \pi_q} \mu_{q,p}(\mathcal{C}_{S_Q}) \quad (2)$$

In this new setting, age and coherency constraints are now seen as *minimal* requirements, and the problem becomes to find a configuration $\mathcal{C}_{S_Q}$ that maximizes the above utility under the age, coherency and resource constraints.

*Example.* In the bandwidth example, all of the queries have the same set of monitoring properties, $\{age, coherency\}$, so the above global utility becomes for this example:

$$\begin{aligned} &\mu_{q_1,a}(C_{S_Q}) \times \mu_{q_1,c}(C_{S_Q}) + \\ &\quad \mu_{q_2,a}(C_{S_Q}) \times \mu_{q_2,c}(C_{S_Q}) + \\ &\quad \mu_{q_3,a}(C_{S_Q}) \times \mu_{q_3,c}(C_{S_Q}) \end{aligned} \quad (3)$$

These utility functions use the same computation to get the age and the coherency of a query, *i.e.,* the age of the query $q$ is given by the minimal frequency among its data sources imposed by the configuration (it is the same for the coherence):

$$a_q = \frac{1}{\min_{s \in \iota_q} C_{S_Q}(f_s)}$$

Adding utility functions $\mu_{q,a}$ and $\mu_{q,c}$ for each query $q_1$, $q_2$ and $q_3$ provides for an optimization problem where the objective is to maximize the equation 3 under the previous age and coherency constraints and the above resource constraint.

Consider the ten data sources and three queries of the previous example, If the total bandwidth $TB = 130kb/s$, the monitoring bandwidth should not to pass over $13kb/s$, from the resource constraint specified in equation (1). In overloaded situation, the QoI requirements are now expressed as utility functions (see Figure 2) to concentrate the resources on the most important queries. In this case, the utility associated to query $q_1$ expresses less stringent QoI requirements on coherency and age than $q_2$ and $q_3$. The following configuration of frequencies maximizes[2] the global utility specified in equation (3), under the age, coherency and resource constraints.

$$\begin{array}{lllllll} f_{hp_1} & = & 2 & & f_{ha_1} & = & 2 \\ f_{hp_2} & = & 2 & & f_{ha_2} & = & 2 \\ f_{rp_1} & = & 2 & & f_{rp_2} & = & 2 \\ f_{tp_1} & = & 1/5 & & f_{tc_1} & = & 1/5 \\ f_{tp_2} & = & 1/5 & & f_{tc_2} & = & 1/5 \end{array}$$

This result shows that the utility leads the monitoring to reduce the QoI for transportation teams, and hence bandwidth for their data sources, since they are used only by the query $q_1$ which has less stringent QoI requirements. □

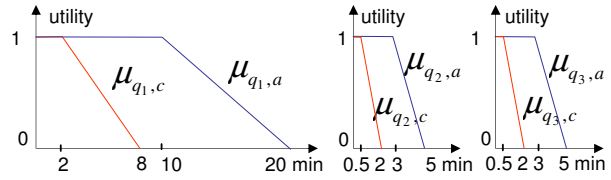



Figure 2. QoI requirements on coherency ($\mu_{q_i,c}$) and age ($\mu_{q_i,a}$), expressed as utility functions for queries $q_1, q_2, q_3$.

## IV. ADAMO Component-based Architecture

We now use the model to present the ADAMO architecture and introduce the various abstractions that enforce QoI needs. We then describe how the framework is implemented and discuss our ongoing work to support reusability and extensibility, notably by using appropriate design patterns.

### A. ADAMO principles

The main goal of ADAMO is to produce centralized monitoring systems that can be located in given points of a distributed architecture[3]. The basic operation supported by

[2]We use Gecode (http://www.gecode.org), a constraint programming toolkit to solve this problem. In this example, 10 data sources and 3 possible frequencies for each data sources generate $3^{10} = 59049$ configurations.

[3]Mastering the deployment of several distributed ADAMO entities is part of future work (see Section V).

 ISBN: 978-1-61208-109-0 

ADAMO is to gather data from distributed sources and store them in a buffer system. These data are then processed prior to being delivered to consumers so that different properties are enforced on requested QoI while obeying to resource constraints. In order to provide a reusable and extensible adaptive monitoring framework, the ADAMO architecture must factor out the common structure and behavior from the monitoring specific parts. Doing so results in software artifacts that can be reused with fewer efforts to design and implement a specific monitoring service. ADAMO thus rests on a *component-based approach*. We detail the resulting design as well as the abstractions made by the framework in the following paragraphs.

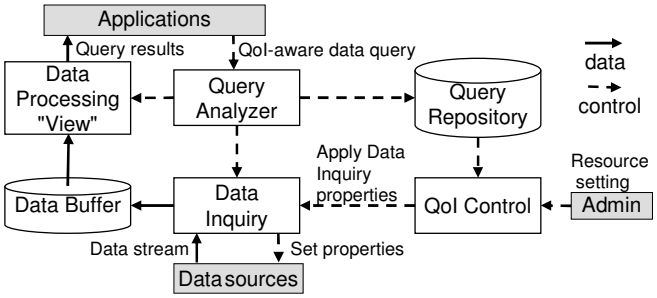



Figure 3. Functional Architecture of ADAMO

At the highest level, the component-based approach allows for structuring the overall architecture of a monitoring system. Figure 3 outlines the main components with their interactions among them and with the external roles described in section II-B.

The *application* represents consumers of the monitoring system, the *query analyzer* acts as the front-end to process different kinds of QoI-aware data queries. For example, applications may fire a batch of queries against the monitoring component and then wait for streaming results, or on the other hand they may submit a query on-demand (in our illustration, before the displacement of inhabitants process starts). The *query analyzer* thus handles queries, initiates *data inquiry* which may derive from a composite dimension and intersect between multiple consumers, identifies consumer's QoI constraints, and stores them into *query repository* for further reasoning. *QoI control* then finds an appropriate configuration for any *data inquiry*. Based on the configuration set, *data inquiry* establishes an inquiry strategy to access remote *data sources*. The inquired data stream is cached in local *data buffers*. Further *data processing view* (called *view* for short) such as QoI filtering or data transformation is realized before delivering final data to the consuming applications, in push or pull mode.

### B. Abstraction of the Framework

In ADAMO various abstraction points are available to clarify domain intents and reduce implementation efforts. This allows software architects to focus on solving a problem without being concerned about less relevant lower level details. In the framework, each component represents a level of abstraction that can be extended to specific adaptive monitoring requirements. For example, *QoI control* can be extended in order to adopt a new trade-off algorithm taking into account coherency and some resource constraints.

*1) Query Analyzer:* A *query analyzer* is in charge of handling and processing data queries. As modeled in III-A, a query consists of two specific parts in which (a) a list of dimensions is used to identify data sources, configure data buffers and views, (b) QoI constraints are used to configure ADAMO, especially data inquiry properties. ADAMO then supports two ways to submit a query: static and incremental. In the static mode all queries are submitted to the monitoring service once and for all. A set of data sources $S_Q$ is then derived from the set of queries $Q$. The incremental mode is obviously more complex as queries can be subscribed and removed at runtime. This requires some specific support on existing queries so that data inquiry processes are correctly deactivated. A new set of data sources is then derived from the pre-existent ones and the new query: $S_Q = f(S_Q^{'}, q)$. In both cases, when multiple clients refer to the same data source, the *query analyzer* makes the necessary adjustments to converge to a single data inquiry, so that duplicated remote data transmissions from data sources are avoided.

Due to the necessary knowledge on data queries for both the query analyzer and the QoI control component, information related to queries, data sources and their relationships is indexed and stored in a *query repository*.

*2) Data Inquiry:* The *data inquiry* component establishes a data inquiry protocol, based on a given configuration $C_{S_Q}$ assigned to data source properties. Data source properties include frequency, message size, data transmission mode (push/pull), but also inquiry mode (batching multiple samples, summary techniques). In practice, message size and data transmission are usually chosen at design time while inquiry frequency is used to regulate data transmission.

*3) Data Processing:* A *data processing view* produces high-level abstract information from some low-level raw data. It also provide the data to the consuming applications according to the protocol of their choice (pull or push mode). In most cases, raw data sensed from environment may be meaningless for consuming applications or some measurements are not good enough for a given QoI request. ADAMO thus distinguishes two types of data processors.

An *Aggregator* aggregates data from different sources to reproduce a new data dimension. A particular case is a translator that transforms data from a unique source. For example, the distance delivered by a data source measured in *mile* can be converted into *kilometer*.

A *QoI-based processor* aims at filtering or evaluating QoI for a given data set. In our motivating example, the

 ISBN: 978-1-61208-109-0 

view representing query $q_2$ (cf. III-B) should filter out position and remaining autonomy of helicopter tuples if they are not coherent in the timing window of 30s. Figure 4 depicts a temporal filter of *<age, coherency>=<2 minutes, 1/2 minute>* that uses a sliding window to select the first coherent tuple of two sources.

In both cases, data processor is fed by data buffers. Multiple consumers hence can share their mutual data sources.

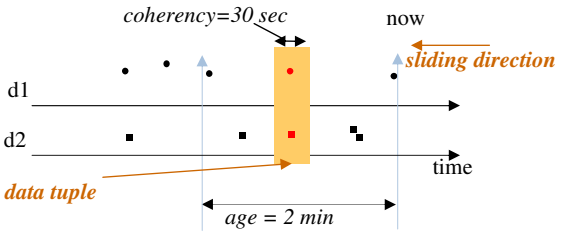



Figure 4. Example - a temporal filter implements QoI based processor.

*4) QoI Control:* A *QoI control* component is used to find a configuration of the monitoring service satisfying QoI requirements and resource constraints. Three distinct tasks are associated to this component. First, it gathers inputs to feed the QoI control algorithms described in section III-B. These inputs consist of knowledge from the query repository (the current set of QoI-aware queries $Q$, the subset of data sources $S_Q$ used by $Q$), and resource settings specified by an administrator (the set of resource constraints $\mathcal{C}_R$). These inputs may vary according to how the QoI control issue is handled. Secondly, it executes the QoI control algorithm to find the data source configuration $C_{S_Q}$ satisfying the current set of queries $Q$ under the resource constraints $C_R$. This algorithm can be changed at run-time thanks to dynamic reconfiguration of components [18]. Finally, it delivers $C_{S_Q}$ to the *data inquiry* component, in charge of applying dynamically this new configuration into the monitoring system.

The configuration of QoI control is typically executed when a new query is submitted. But executing this on-demand is potentially costly. To tackle this issue, ADAMO proposes two strategies for the administrator. First, it proposes two reconfigurations modes[4]: reconfigure all data sources or reconfigure only inactive data sources. Secondly, in ADAMO, it is possible to specify when the reconfiguration are effectively run, based on time-interval (e.g., every 5 minutes) or query unit interval (e.g., every 3 query updates).

### C. Implementation and Reuse of the Framework

The prototype of ADAMO has been implemented to a large extent on top of COSMOS [11], a probe framework for managing context data in ubiquitous applications. This enables the framework to easily reuse many data sources through dedicated wrappers, which are also easy to write or to partly generate. As for its component model, ADAMO relies on the Fractal [19] generic component model, which notably provides hierarchical decomposition of components at any level, explicit definitions of required and provided interfaces, as well as full capabilities for dynamic reconfigurations. Building on this rich component model enables software architects to more easily reuse and/or tailor components inside the framework.

Besides several design patterns are used to improve the design, reuse and consistency of the ADAMO architecture. A typical monitoring system instantiated from ADAMO should implement components by extending the abstraction mechanisms described in the previous section. At the highest level, these components must be consistent with each other and the *Abstract Factory* pattern is then used to ensure this consistency constraint. For example, to tackle a new QoI concept as first-class constraint such as data precision of query results, one should ensure that the concept is taken into account by every concerned monitoring entities, i.e., *query analyzer*, *data inquiry*, *view* and *QoI control*. Listing 1 shows an excerpt of Abstract Monitoring Factory interface.

Listing 1. Abstract Monitoring Factory

```
public interface AbstractMonitoringFactory {
  public QueryAnalyzer createQueryAnalyzer();
  public DataInquiry creatDataInquiry();
  public View creatView();
  public QoIControl creatQoIControl();
}
```

As building a monitoring service implies creating a set of ADAMO components, the *Composite* pattern is reused in the architecture to support two specific compositions. The composition capability of *View* extends the composition provided by the COSMOS probe system so that a data access point dedicated to a data query is assembled from data inquiries, data buffers and data processors. ADAMO composition adds *query analyzer*, *query repository*, and *QoI control* into each ADAMO instance to enable the QoI control capabilities. These compositions rely on Fractal ADL [19]. Figure 5 illustrates this organization with three queries described in III-B.

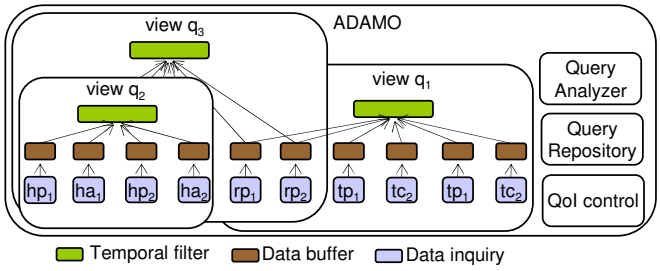



Figure 5. Composition example.

As multiple consumers may be interested in the same source, data transmission is improved by creating a single transmission channel between ADAMO and every needed

[4]The design of other reconfiguration modes is part of future work.

 ISBN: 978-1-61208-109-0 

data source. While the QoI Control component configures the mutual data inquiry to satisfy different requests, the *Flyweight* pattern enables the reification of the view composition so to data buffers are shared between consumers. As illustrated in Figure 5, data buffers of rescue teams $rp_1$ and $rp_2$ are included in both views of queries $q_1$ and $q_3$.

Finally, the *Singleton* pattern ensures that each ADAMO instance has only one *query analyzer*, *query repository* and *QoI control*. The *Query analyzer* then provides a global point of access to consumers (acting as *Facade* pattern), whereas the latter two maintain coherency on monitoring constraints and monitoring algorithms.

## V. Conclusion and Future Work

In this paper, we have presented ADAMO, an adaptive monitoring framework that tackles different QoI-aware data queries over dynamic data streams. The proposed system relies on a constraint-solving approach and component-based techniques so that common structures and behaviors of monitoring systems can be more easily reusable and extensible. We have shown how it provides solutions to handle multiple clients with different QoI requirements, transformation of QoI needs into probe configuration settings, control trade-offs between QoI needs and resource constraints, and management of data queries in a static or incremental way.

Regarding future work, short term goals are to evaluate the effectiveness of the proposed framework with stress/load testing and to validate its genericity with different scenarios and more QoI dimensions, including precision and significance as proposed in [1], [9]. In the long term, we plan to tackle scalability issues by providing self-regulation capabilities and by enabling several ADAMO monitoring systems to be distributed.

## Acknowledgment

The research was partly funded by the French National Research Agency (ANR) through the SemEUsE research project. See http://www.semeuse.org.

## References

[1] T. Buchholz, A. Kupper, and M. Schiffers, "Quality of context information: What it is and why we need it," in *Proceeding of the 10th HP-OVUA Workshop, Geneva, Switzerland*, 2003.

[2] J. Joyce, G. Lomow, K. Slind, and B. Unger, "Monitoring distributed systems," *ACM Trans. Comput. Syst.*, vol. 5, no. 2, pp. 121–150, 1987.

[3] L. Baresi, S. Guinea, and P. Plebani, "WS-Policy for service monitoring," in *Technologies for E-Services, 6th International Workshop, TES 2005, Trondheim, Norway, September 2-3, 2005, Revised Selected Papers*, ser. LNCS, vol. 3811. Springer, 2006, pp. 72–83.

[4] F. Barbon, P. Traverso, M. Pistore, and M. Trainotti, "Run-time monitoring of instances and classes of web service compositions," in *Web Services, 2006. ICWS '06. International Conference on*, Sept. 2006, pp. 63–71.

[5] O. Moser, F. Rosenberg, and S. Dustdar, "Non-intrusive monitoring and service adaptation for ws-bpel," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, USA: ACM, 2008, pp. 815–824.

[6] Z. Abid, S. Chabridon, and D. Conan, "A framework for quality of context management," in *QuaCon*, 2009, pp. 120–131.

[7] N. Jain, P. Yalagandula, M. Dahlin, and Y. Zhang, "Self-tuning, bandwidth-aware monitoring for dynamic data streams," in *ICDE '09: Proceedings of the 2009 IEEE International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 114–125.

[8] M. A. Munawar and P. A. S. Ward, "Adaptive monitoring in enterprise software systems," in *SIGMETRICS 2006 Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (SysML)*, 2006.

[9] A. Manzoor, H.-L. Truong, and S. Dustdar, "On the evaluation of quality of context," in *EuroSSC '08: Proceedings of the 3rd European Conference on Smart Sensing and Context*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 140–153.

[10] M. Anwar Hossain, P. Atrey, and A. El Saddik, "Context-aware qoi computation in multi-sensor systems," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, Oct 2008, pp. 736–741.

[11] D. Conan, R. Rouvoy, and L. Seinturier, "Scalable processing of context information with cosmos," in *DAIS*, 2007, pp. 210–224.

[12] V. Poladian, J. P. Sousa, D. Garlan, and M. Shaw, "Dynamic configuration of resource-aware services," in *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 604–613.

[13] V. Poladian, D. Garlan, M. Shaw, M. Satyanarayanan, B. Schmerl, and J. Sousa, "Leveraging resource prediction for anticipatory dynamic configuration," in *SASO '07: Proceedings of the First International Conference on Self-Adaptive and Self-Organizing Systems*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 214–223.

[14] B. Babcock, M. Datar, and R. Motwani, "Load shedding for aggregation queries over data streams," in *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*. Washington, DC, USA: IEEE Computer Society, 2004, p. 350.

[15] N. Tatbul, U. Çetintemel, and S. Zdonik, "Staying fit: efficient load shedding techniques for distributed stream processing," in *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*. VLDB Endowment, 2007, pp. 159–170.

[16] M. A. Munawar, M. Jiang, and P. A. S. Ward, "Monitoring multi-tier clustered systems with invariant metric relationships," in *SEAMS '08: Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*. New York, USA: ACM, 2008, pp. 73–80.

[17] S. Agarwala, Y. Chen, D. Milojicic, and K. Schwan, "Qmon: Qos- and utility-aware monitoring in enterprise systems," in *Autonomic Computing, 2006. ICAC '06. IEEE International Conference on*, June 2006.

[18] P.-C. David and T. Ledoux, "Safe dynamic reconfigurations of fractal architectures with FScript," in *the 5th Fractal Workshop at ECOOP 2006*, Nantes, France, Jul. 2006.

[19] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J.-B. Stefani, "The fractal component model and its support in java: Experiences with auto-adaptive and reconfigurable systems," *Softw. Pract. Exper.*, vol. 36, no. 11-12, pp. 1257–1284, 2006.

 ISBN: 978-1-61208-109-0 

# Tracing Structural Changes of Adaptive Systems

Kai Nehring
*AG Software Engineering: Dependability*
*University of Kaiserslautern*
*Kaiserslautern, Germany*
*Email: nehring@cs.uni-kl.de*

Peter Liggesmeyer
*AG Software Engineering: Dependability*
*University of Kaiserslautern*
*Kaiserslautern, Germany*
*liggesmeyer@cs.uni-kl.de*

***Abstract*—The internal structure of an adaptive system is subject to change. An unforeseen and unwanted component composition can cause serious malfunction in a system, which can be difficult to track. Knowledge of the system's internal structure at runtime helps to understand the system and, in the end, helps to test a system more thoroughly. In this paper, we present the provisional result of our ongoing research on an approach that tests the reconfiguration of adaptive systems.**

***Keywords*-adaptive system; BTrace; component composition; internal structure; state tracking**

## I. Introduction

In a demanding world where systems not only have to be available incessantly but also have to adapt to changing environments and/or requirements, (dynamically) adaptive systems gain popularity. Testing such systems is still a challenging task. Several approaches have been published over the years, such as [1][2][3] to name a few. Most approaches test the functionality of an adaptable system but not the quality requirements, such as elapse time of the adaptation or the states which occur during the reconfiguration.

Other approaches take some of these requirements into account [4][5] but often require specialised knowledge and complicated system descriptions, often in terms of temporal logic (A-LTL) [6][7] which can be a challenging task, too.

In our work, we focalise on a test approach for quality requirements of adaptive systems that uses information which are already available, such as component diagrams, and a methodology which does not need in-depth knowledge beyond that of an average system developer.

Our current work is concerned with the internal structure of a system and the representation of changes which occur at runtime. These information can be used to gain knowledge of the system, e.g., the individual states of an application during reconfiguration. Furthermore, these data can be used to evaluate the architecture by applying metrics [8] and eventually to improve the system's design.

In Section II, we present a general approach to track structural changes. Section III introduces the toolset we use to track changes in Java applications. We also applied the suggested approach to a demonstration system and present the result of that experiment in Section IV. Section V gives an overview of the future work.

## II. Tracing structural changes

A structural change appears if one component is replaced with another. Several steps are usually necessary to replace a component. At least all references which point to that component have to be reset in order to point to the replacement. This can require many operations, depending on the number of references which have to be updated. Often a dedicated controller is responsible to reconfigure the system, e.g., a component manager or a configuration manager. Various information can be of interest, depending on the point of view, such as:

- How long did the reconfiguration take?
- What components are currently (or previously) connected with the component of interest?
- What configurations passed the system through the reconfiguration?

The reconfiguration elapse time can be of great importance whenever time constrains have to be fulfilled, e.g., the reconfiguration elapse time for an ordering system must not exceed 1500 ms. Since the reconfiguration of a system is usually done within a set of methods, the elapse time of these methods have to be recorded. During application tracing, the time stamp of both the entry-point and return-point of a method have to be recorded in order to calculate the elapse time. Depending on the structure of the system, multiple methods might have to be traced and elapse times have to be summarised.

To answer the second question, an approach is necessary that allows to add probes into the runtime system which fire whenever an attribute of interest will be changed, e.g., references to components which are likely to be replaced at runtime. It is important to observe all components which use these replaceable components in order to track configuration changes. A recorded change (delta set) comprises the

- source of the change, i.e., name of the component and its identification hash code
- attribute that held the reference and then holds the new reference after the modification
- target reference, i.e., name of the new component and its identification hash code
- time of occurrence when the change took place

 ISBN: 978-1-61208-109-0 

A probe is a piece of code that will be executed whenever a certain event occurs. This can be

1) a method in the original code, i.e., additional code in a set-Method,
2) an aspect that has been woven around an attribute/method or
3) code that has been injected into the runtime system.

The probe has to log the original value (before-value) and the new value (after-value). The change in value can be used to track configuration changes: a reference of a component now points to a different component; therefore, the configuration has changed.

That leads to the answer to the third question. To identify and eventually analyse the different states during the reconfiguration, all delta sets have to be applied on the initial state (before the reconfiguration) of the system in order of occurrence, i.e., the final state of the system is built upon the initial state and the delta sets. Based on that, evaluation methods can be applied to find illegal states, e.g., to mark all occurrences of components which hold a reference to an outdated component.

The previously mentioned approach requires the state of the system right before the reconfiguration takes place, i.e., the initial state. Although it is possible to trace an application right from the start, it is often impossible or at least very difficult, especially if an application shall be observed under real-life conditions. The initial state can be calculated, too, by using the before-values of the recorded changes. A before-value correlates with the association to another component before it is going to change. Both the association and the associated component must be preserved in order to calculate the initial state. By iterating over all changes, the state will be built up. Attributes with value `null`, however, must be ignored. Components that are newly created usually have all attributes set to `null`. In that case, the whole component has to be ignored since it was not part of the composition before. The remaining connections among components have to be summed up in order to create the previous state right before the first change. This state, however, does not include components, which hold references to one of the displayed components, that do not anticipate in the reconfiguration process, i.e. the state might be incomplete.

## III. Recording changes with BTrace

We use BTrace [9] to trace changes at runtime. BTrace allows to insert probes into the Java Virtual Machine and therefore to observe applications even if their source code is not available by using the Java Instrumentation API. Applications do not need to be modified or recompiled in any way.

BTrace offers a set of probes which fire on certain events, such as entry or return of a method or whenever an observed attribute will be modified. It also allows to record the values of attributes before and after the manipulation. A probe typically consists of an action method which should be executed if a certain event occurs and an annotation which tells BTrace when to fire.

The action method is similar to a Java method but its functionality is reduced to a minimum. It is not possible to overwrite attributes of the application nor to create new objects. BTrace also prohibits loops and some long running operations. It is basically an observation tool designed for minimal impact.

The annotation of an action method tells BTrace what to observe, when to fire and what information to collect. The following annotation causes BTrace to fire whenever the method reconfig in class pwgen.Main is executed. An action method annotated that way would be executed right before (Kind.ENTRY) the execution of the reconfig-method.

```
@OnMethod(clazz="pwgen.Main",
   method="reconfig",
   location=@Location(Kind.ENTRY))
```

The annotation that is necessary to trace field manipulation is insignificantly more complex. Please note, that a wildcard is used to declare the method tag since all manipulation attempts should be tracked regardless which method causes the manipulation of an observed attribute.

```
@OnMethod(clazz="pwgen.Main",
   method="/.*/",
   location=@Location(
      value=Kind.FIELD_SET,
      clazz="pwgen.Main",
      field="pwGen",
      where=Where.BEFORE))
```

A special compiler compiles the script and BTrace will inject the resulting class into the JavaVM for a defined Java process. It is therefore possible to observe several Java processes in parallel, each with a different script. BTrace currently exists as a console application and as an add-on for VisualVM [10]. Latter offers an easy to use graphical user interface.

## IV. Trace of a Demo-System

We applied the approach on a demonstration system which is used to teach adaptive behaviour in software. Following the procedure, we have found a so far unknown defect in one of the software components which could cause a failure.

The system under test was a password generator which is comprised of the following components:

**SRG:** A SimpleRandomGenerator that produces random numbers. It can be connected to only one client at a time.

**PwGen2:** Password generator with reduced character set (literal only). For further speed optimisations, the length of the password is limited to 6 characters.

 ISBN: 978-1-61208-109-0 

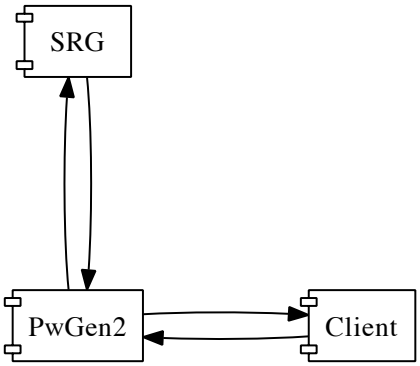



Figure 1. Configuration 1 (initial state): fast, but only suitable for weak passwords

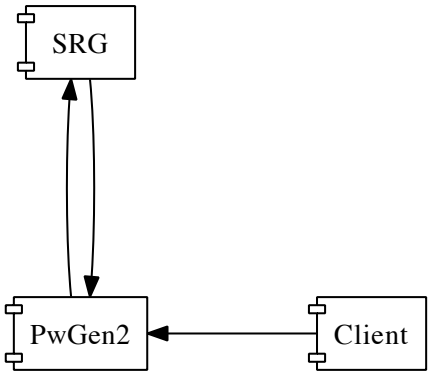



Figure 2. First step of the reconfiguration: Connection between PwGen2 and Client was removed

**PwGen1:** Enhanced password generator that uses an extended character set (literal, digit and special character) and offers a more liberal limitation of the password length. The enhanced feature set results in lower execution speed.

**Client:** The client uses a password generator, either PwGen2 (default) or PwGen1 (if strong passwords are requested).

Please note that each component comprises its own component manager, not shown in the figures. The component manager sets up each component if necessary and connects it with required services, e.g., the component manager of PwGen2 requests the service of SRG and injects its reference into PwGen2 if SRG is available. It also releases SRG if its service is no longer needed.

Method calls are performed asynchronous. Each component therefore requires not only a reference to a service provider but also needs to inject its own reference to the provider in order to receive the provided service, depicted in Figure 1.

The requirements on the password strength changes at runtime. The generator reconfigures itself to use PwGen1 instead of PwGen2. Several steps are necessary to replace this component:

1) Client: remove client reference in PwGen2 that points to the Client component
2) PwGen2: release SRG, i.e., the client reference of SRG that points to PwGen2 has to be removed
3) PwGen2: release reference to SRG
4) PwGen2: release reference to Client
5) Client: establish connection to PwGen1 and inject reference to Client
6) PwGen1: establish connection to SRG and inject reference to PwGen1
7) SRG: establish connection to PwGen1 and inject reference
8) PwGen1 (now fully functional): inject reference into Client

In the end, the composition of components should look like Figure 1 but with PwGen1 in place of PwGen2.

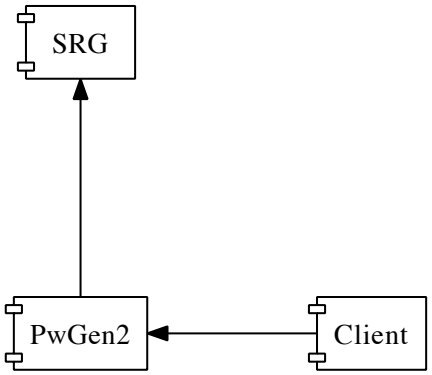



Figure 3. PwGen2 releases SRG; SRG can now be used by another component

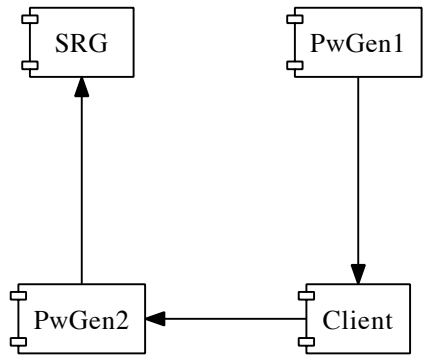



Figure 4. Client requests service of PwGen1 and injects its reference

We added probes to trace important events, such as changes in component connections and ran the demo application. We then analysed the trace result and generated a visual representation of the component composition before, during and after the reconfiguration, depicted in Figure 2 to Figure 7.

The result of the reconfiguration, depicted in Figure 7,

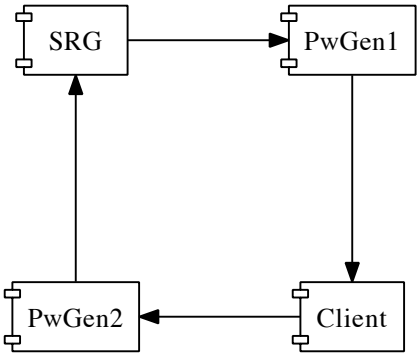



Figure 5. PwGen1 requests service of SRG and injects its reference

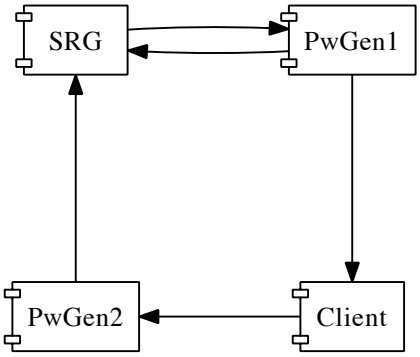



Figure 6. SRG accepts PwGen1's service request and injects its reference to PwGen1

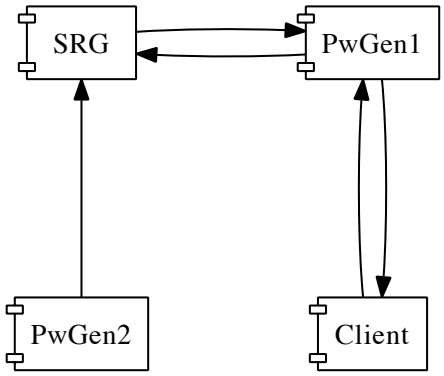



Figure 7. Configuration 2 (final state): resulting composition after reconfiguration (PwGen1 has injected its reference to Client)

derives from the expected result. It turned out that the component manager of PwGen2 did not release the reference to SRG. The resulting composition can cause serious malfunction if PwGen2 is used accidentally. In that case, PwGen2 would request a new random number from SRG. The request would not fail since the reference to SRG is still valid. SRG would generate a new random number but would send it to PwGen1 since its client reference points to PwGen1, which doesn't expect a new number. The impact of this event depends on the implementation of PwGen1. In our case, it added a new character to the current password and, if the minimum password length was (already) reached, sends the password to the Client. The client overwrote the old password with the new one although it didn't request that password.

## V. Conclusion and Future Work

As the demonstration shows, serious malfunctions can appear due to erroneous composition of components. Misbehaviour can be difficult to track because its origin was a component that ought to be not active anymore. Knowledge of the actual component composition can help to understand certain failures and can help to identify defects. Furthermore, it can help to create a new class of test cases to test a system more thoroughly.

In future work, we will define metrics and a methodology to determine whether a reconfiguration process satisfies given quality requirements, such as reconfiguration elapse time or latency caused by the adaptation. Target of our research will be the adaptation process itself.

## References

[1] X. Bai, Y. Chen, and Z. Shao. Adaptive web services testing. *Computer Software and Applications Conference, Annual International*, 2:233–236, 2007.

[2] J. Grundy, G. Ding, and G. Ding. Automatic validation of deployed j2ee components using aspects. In *In Proc. 2002 IEEE International Conference on Automated Software Engineering*, pages 47–58. IEEE CS Press, 2001.

[3] component+ Partners. Built-in testing for component-based development. *EC IST 5th Framework Project IST-1999-20162 Component+*, Technical Report D3, 2001.

[4] H. J. Goldsby, B. H. Cheng, and J. Zhang. Amoeba-rt: Runtime verification of adaptive software. pages 212–224, 2008.

[5] K. N. Biyani and S. S. Kulkarni. Assurance of dynamic adaptation in distributed systems. *J. Parallel Distrib. Comput.*, 68(8):1097–1112, 2008.

[6] J. Zhang and B. H. C. Cheng. Using temporal logic to specify adaptive program semantics. *Journal of Systems and Software*, Volume 79(10):1361–1369, 2006.

[7] J. Zhang, H. J. Goldsby, and B. H. Cheng. Modular verification of dynamically adaptive systems. In *AOSD '09: Proceedings of the 8th ACM international conference on Aspect-oriented software development*, pages 161–172, New York, NY, USA, 2009. ACM.

[8] C. Raibulet and L. Masciadri. Evaluation of dynamic adaptivity through metrics: an achievable target? In *WICSA/ECSA*, pages 341–344, 2009.

[9] http://kenai.com/projects/btrace 06.18.2010

[10] https://visualvm.dev.java.net 06.18.2010

 ISBN: 978-1-61208-109-0 

# Quantifying Adaptability

**Ken Krechmer**
Lecturer, University of Colorado
Palo Alto, CA 94303-3024 USA
krechmer@csrstds.com

***Abstract*—Until adaptability can be calculated and measured, it is difficult to understand or promote. This paper defines adaptability and offers a way to measure it based on communications theory. Adaptability is a complex process which requires a number of supporting processes and definitions. The concept of state-pairs is developed and used to describe communicating entities, interfaces, comparison, measured information, communications, flexibility and, finally, adaptability. Interfaces are delineated by their percent of adaptability; different means are identified to add the benefits of adaptability to existing and future systems.**

***Keywords-adaptability; interface; communications structure; measured information; etiquette***

## I. INTRODUCTION

This paper defines *adaptability* as the process of automatically negotiating possible parameters, as it makes a system more adaptable. Some assumptions in this definition include: a negotiation occurs between at least two entities, and a negotiation requires communications. With the recognition that communications is a requirement before adaptability can be realized, adaptability is defined based on communications theory. First the structure of a communications system used to transfer a message between transmitter and receiver entities is developed. From this structure an interface is derived. Defining an interface allows an understanding of how to adapt and measure an interface. Quantifying adaptability also adds to the understanding of how to implement adaptable systems.

## II. A COMMUNICATIONS STRUCTURE

Adaptability requires communications. Fig. 1 models communications for the purpose of understanding its structure rather than its performance. Fig. 1 is similar to the Shannon model of a communications system except that the communications channel is replaced by an interface and the probability of the output message being the same as the input message is fixed to one [1, page 34]. The transmitter (T) and receiver (R) are communicating entities connected via an interface. This model allows an analysis of the structure of the relationship between T and R.

From communications theory, T and R support all the state-pairs $t_i$ - $r_i$, where i includes the set of all t or r states 1 to n in Fig. 1. A state-pair includes a specific input part ($t_x$) associated with T, which is related to the output part ($r_x$) associated with R.

The relationship between $t_x$ and $r_x$ is described as one-one. "A relation is said to be one-one when, if t has the relation in question to r, no other term t' has the same relation to r, and t does not have the same relation to any other term r' other than r" [2]. All the state-pairs associated with T and R form the interface between T and R. A single set of $t_i$ or $r_i$ states is termed a parameter of the transmitter or receiver. Communications between T and R (information transfer) is possible only when multiple state-pairs form an interface. Most interfaces include multiple parameters. An interface is a concept that does not exist independently, it is only formed by the common parameters of the related entities.

A state-pair is in some ways a relationship between equal elements. As example in Fig. 1, state $t_x$ may be considered as the equal of state $r_x$. However, it is not possible to define such equality exactly without specifying other sets of state-pairs. For state $t_x$ to be equal to $r_x$ for example, the boundary conditions of each state (defined by other state-pairs) must be equal. To avoid this complexity, this paper considers a state-pair to be one-one, not necessarily equals.

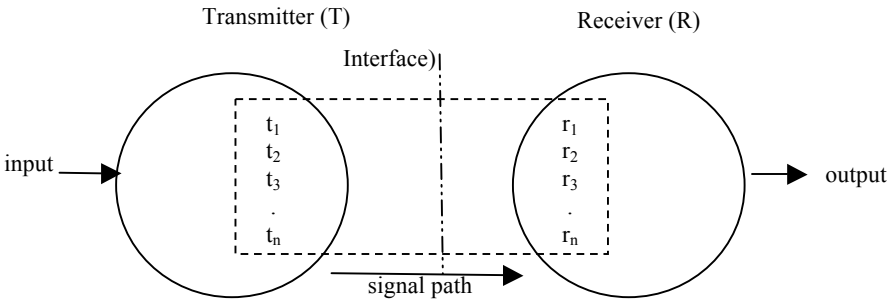



Figure 1. Communications structure.

 ISBN: 978-1-61208-109-0 

The concept of state-pairs may be applied to any interface, even a physical interface. Examine a perfectly compatible (zero tolerance) physical plug and socket. The outside diameter of the plug and the inside diameter of the socket are the same. The length of the plug and the socket are the same. The interface between the plug and socket consists of all the points on the same diameters along the same length of the socket and plug. These common points are the state-pairs which form a physically compatible interface. In a real physical interface, multiple layers of sets of state-pairs are needed to completely define the interface including: the physical dimension system used and the tolerances applied.

## III. COMMUNICATIONS PROCESS

The communications structure in Fig. 1 allows a detailed view of a communications process across the interface. This is necessary to see the how state-pairs create an interface and eventually to use the number of state-pairs to quantify adaptability.

The ability to pass information across each state-pair requires two comparisons. Each comparison is associated with a part of a state-pair. The fundamental nature of these comparisons is suggested by I. Kant who states that a comparison is necessary for understanding [3].

TABLE I. COMMUNICATIONS PROCESS

| | |
|---|---|
| **For the transmitter (T):** | |
| 1 | Select an input message |
| 2 | Compare this input and determine state ($t_i$) |
| 3 | Output a signal |
| **For the receiver (R):** | |
| 4 | Select the signal received |
| 5 | Compare this signal and determine related state ($r_i$) |
| 6 | Output message |

The simplest communications process may consist of six operations, three in the transmitter and three in the receiver (Table 1). What is critical to this analysis is that the communications process consists of symmetric transmit and receive processes, each of which includes a comparison. The symmetry about the interface first appears in the concept of one-one and also appears in Fig. 3 (below). Operations 2 and 5 in Table 1 demonstrate how a state-pair relates to the communications process.

Consider a binary amplitude-modulated communications system with two state-pairs ($t_1$ - $r_1$ and $t_2$ - $r_2$) and without time domain or tolerance effects. The input message to T is compared with the decision boundary between $t_1$ and $t_2$ determining which state causes a T signal output in volts (V). T encodes +V signals for $t_1$ and -V signals for $t_2$ which are received as signals in R. The received signal is compared with the decision boundary between $r_1$ and $r_2$ determining which state causes a R output message. R decodes +V signals to $r_1$ and -V signals to $r_2$. The decision boundaries, both threshold and maximum, are lower level parameters (formed by other parameters created in the implementation of T and R). These boundaries implement the relationship between each part of the $t_1$ - $r_1$ and $t_2$ - $r_2$ state-pairs and determine the operational characteristics of the signal path. A more complex communications system has more sets of transmitter and receiver state-pairs (parameters) and more complex boundaries.

Example: In the course of reading, a word appears of unknown meaning. The reader refers to a dictionary. A dictionary relates words (states) to their meanings (message). The author and reader select words from similar dictionaries (first and second comparisons). The author's and reader's dictionaries together are the state-pairs of equal words with a common meaning in each dictionary.

The state-pairs in a communications system may be created by chemical bonds (A-C, G-T in DNA), pre-existing written or spoken alphabets, pre-existing dictionaries or syntax, the specifications or standards defining a transmitter, receiver or protocol (electronic communications) or a physical implementation of a transmitter, transmission link, or receiver. Different functionalities of state-pairs are divided into layers in the Open System Interconnect model (OSI) where each reference layer provides the interface(s) used by the next layer. Layer one includes physical aspects of the interface and higher layers include successively more abstract functionality.

## IV. MEASUREMENT PROCESS

With a model of a communications structure and communications process, a measurement can be defined in mathematical terms. When a measurement is understood, communication entities and their interface may be defined. Then adaptability can be measured based on the interface between the communicating entities.

A measurement is a quantified selection from an entity or process (E/P) being measured. Making a quantified selection is similar to the communications process shown in Table 1. In a measurement process there is a receiver with a set of states used for comparison with the signal from an E/P to be measured. The signal from the E/P may be generated by the E/P or applied externally to the E/P to generate a reflected signal. This signal from the E/P may be seen as from a transmitter. The receiver must have one-one states with the signals received for a practical measurement. The similarity of a measurement and a communications process supports the use of communication theory to analyze the measurement process.

N. Campbell defines a measurement (the concept) as "the assignment of numerals to represent properties" [4]. A measurement process assigns the numerals by utilizing one or more comparisons between the signal received and the states of the receiver. Each of the states of the receiver, and its associated boundary conditions, acts to quantify the measurement. Any parameter (property) of an E/P which may be quantified, e.g., weight, length, color, hardness, texture, transfer rate, capacity, spin, etc. may be a measured parameter.

The choice of the receiver, its range of states and boundary conditions actually selects and quantifies the parameter of the E/P to be measured. That is, if a length scale is used, distance is measured; if a weight scale is used, weight is measured; if a voltmeter is used, voltage is measured, etc. A measurement is not absolute; it is always

 ISBN: 978-1-61208-109-0 

relative to the parameter measured by the receiver, the states of that parameter in the receiver and the boundary conditions between states. This requires that the states of the receiver be represented in a definition of measured information.

## V. DEFINING MEASURED INFORMATION

The following paragraphs develop the mathematical basis for the information contained in the measurement of a parameter (T) of an E/P. When the quantity of measured information transmitted across an interface is known then the adaptability of that interface may be quantified.

$$H(T) = -\sum_{i=1}^{i=n} p(t_i) \log p(t_i) \quad (1)$$

$$D_t = \log n + \sum_{i=1}^{i=n} p(t_i) \log p(t_i) \quad (2)$$

Equation (1) is Shannon's equation for entropy [1, page 50]. *p* describes the probability of *(x)*. $D_t$ (2) is defined in T. Cover and J. Thomas as the entropy relative to *log n* [5, page 27]. This following paragraphs develop a proof that $D_t$ represents the information contained in the measurement of a parameter (T) of an E/P.[1] A receiver (for $t_i$) with *n* discrete states is assumed in (2). The *n* discrete states are represented by the first term (*log n)* in (2). The entropy distribution (*H(T)*) of the measurement process is calculated by the second term. The first term is the reference applied to quantify the second term. The output from the measurement process is one or more specific states $t_x$, $t_y$, $t_z$. The measured information is equal to $D_t$.

As example, when a voltmeter (used to measure volts) with a 3 volt full scale (parameter of the voltmeter) and the minimum measurable increment (tolerance) of 0.1 V, has 30 (= n) possible states of $v_i$ and produces a single output measurement $v_x$, then $D_v$ = log 30. The greater the number of states n, the greater the information from the measurement process. The narrower the distribution of the entropy term (*H(T)*), the greater the information. A perfect measurement (zero *H(T)*) produces the maximum information, *log n*. The first term of (2) effectively includes the concept of tolerance (minimum measurable increment) in the measured information calculation.

Fig. 2 expresses (2) as a Venn diagram. Fig. 2 shows how the limit of the entropy distribution (*log n*) is related to the entropy distribution (*H(T)*). Equation (3) is Cover and Thomas' equation for Mutual Information (*MI*), the relative entropy between related entropy distributions. Replacing *H(R)* in (3) [5, page 19] with *log n* calculates the mutual information of *H(T)* and *log n* (4).

$$MI = I(T;R) = H(R) - H(R|T) \quad (3)$$

Expand $H(R|T)=H(R)-H(T)$, then replace *H(R)* with *log n*

$$MI = log\ n - (log\ n - H(T)) \quad (4)$$

[1] This definition of measured information has been proposed before. The first time may be in 1957 [6]. To the author's knowledge this paper offers the first proof of this definition.

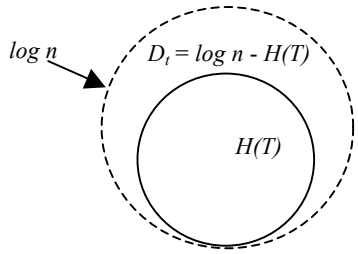



Figure 2. Venn diagram of *H(T)* and its limit.

$$MI = H(T) = I(T; log\ n) \quad (5)$$

Equation (5) shows that *H(T)* when referenced to its limit is equal to the mutual information as the *log n* terms in (4) cancel each other. Using $D_t$ (2) provides a rigorous description of measured information without changing mutual information (MI).

In support of using $H(R|T) = H(R) - H(T)$ above, a related result to (5) substitutes *H(T)* for *H(R)* in (3) and is noted as self-information [5, page 20]. Equation (5) and self-information indicate that the reference may be either *log n* or *H(T)* itself. If the reference is not *log n* or *H(T)* itself, then there are additional parameters (not T). A single parameter entropy distribution should be referenced to its limit (i.e., *log n*), as applying *H(T)* to reference *H(T)* is self-referential. The measured information related to a single parameter entropy distribution only exists in relation to a reference and the only logical reference is the limit of the entropy distribution. This provides a proof of $D_t$ (2) as the definition of measured information.

Consider a measurement of the length of an entity using a meter scale. The meter scale is divided into 1000 increments. The entity is the same number of increments long (*x*) every time it is measured, i.e., the entropy distribution of this measurement is zero (log 1 = 0). The measured length is *x* (the data). Then the information contained in the measurement is log 1000. The quantity of measured information is the same for any accurate measurement using the same meter scale (reference).

## VI. COMMUNICATIONS

Communications across an interface makes adaptability possible. This requires the six operations from Table 1 (above). These six operations occur across the interface formed by the state-pairs.

A communications system is modeled by using two overlapping Venn diagrams from Fig. 2 as shown in Fig. 3. The second Venn diagram in Fig. 3 models *H(R)* and $log\ n_r$ and their relationship $D_r$. Fig. 3 is derived from Shannon's model of a communications system, where the receiver output is related to the transmitter input by a probability less than one. Fig. 3 models equation (3) with the addition of the limits of *H(T)* and *H(R)* and their associated intersection.

In Fig. 3 $log\ n_t$ is the bound of *H(T)* and $log\ n_r$ is the bound of *H(R)*. The intersection of log $n_t$ and log $n_r$ is

shown as a dotted lens shape. This intersection represents the interface (*I*) made up of all the possible state-pairs of T and R. *I* limits the *MI* (intersection of the solid circles, solid lens shape) possible between T and R. *MI* represents the entropy distribution transferred from T to R. *I* is the limit of *MI* in the same manner that *log n* is the limit of *H(T)* as shown in Fig. 2. Similarly, the intersection of $D_t$ and $D_r$ (not delineated in Fig. 3) represents the measurement of the information that is communicated, in the same manner that $D_t$ (2) represents the information in a measurement. When *I* (the state-pairs available for communications) is changeable, adaptability is possible.

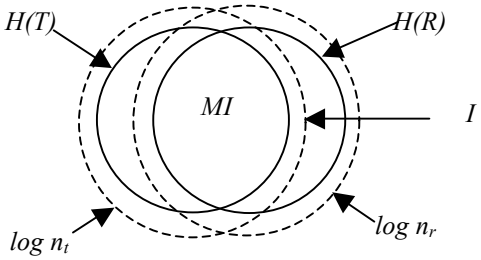



Figure 3. Venn diagram of a communication system.

By expanding on the previous example of the measurement of the length of an entity with a meter scale, a rudimentary communications process may seen. After the first measurement (with first comparison), the data (*x*) is transferred (communicated) to a second entity by applying the meter scale to a second entity and identifying where *x* appears (second comparison). The two meter scales (or one used twice) form the set of state-pairs in this communications process which pass the data (*x*) from the first entity to the second entity. This example shows that a measurement transfer system transfers data by comparing the initial entity to its reference and comparing the referenced data to the second entity.

Fig. 3 offers a similar view to the simple measurement transfer example above, where the transmitter entropy distribution (a measure of the transmitted data distribution) is compared to the common state-pairs (interface) formed by the limits of the transmitter and receiver entropy distributions. This creates the MI which, with unpaired states of the receiver, provides the received entropy distribution (a measure of the received data distribution). This view explains how communications occurs, not by an *H(T)* to *H(R)* interaction, but via the existence of a state-pairs interface.

## VII. ADAPTABILITY

In a communications system, the parameters are defined by sets of state-pairs which form the interface (*I*) between two compatible E/P. The interface (*I*) allows comparisons which can support communications. Changing *I* is what allows adaptability. Using this model it is now possible to define and quantify adaptability.

### A. Unpaired States

Each parameter presented across an interface consists of a number of state-pairs (*n*). However, the number of states in T may not be the same as the number of states in R for some parameters. This situation was not considered in Shannon's work, perhaps because it was seen as a design error. However, in complex communication systems such unpaired states occur when parameters, by virtue of options, special features, differing revisions or just non-selection in the transmitter or receiver, are not available or not used. For instance, telephone modems may support up to six different modulations ranging from 300 bit/s to near 56 kbit/s. Using a standardized protocol (V.8), the telephone modems identify which modulations are available at each end and select the modulation which supports the highest common data rate; the remaining modulations are unused.

Fig. 3 shows unpaired states within each dotted circle areas (*log* $n_t$ and *log* $n_r$) and outside *I*. The communications structure would be more efficient if unpaired states didn't exist. Older communications systems, which tended to be single provider (e.g., telegraph and telephony), tried to avoid unpaired states to be more efficient. But the need for tolerance in older analog receivers always required the receiver states (e.g., bandwidth) be greater than the transmitter states.

Newer communications systems tend to have more and more unpaired states as communications becomes more complex and multi-vendor. This may also be seen as a greater desire for peer-to-peer systems (which will have unpaired states). Interconnected systems have become larger, multi-vendor, and include many revision levels and multiple technologies. Since memory and programming costs are very low and continue to decline, the trend towards less efficient systems (with more unpaired states) appears to be continuing.

At least two approaches have been used to avoid the effects of unpaired states: 1) the selection of other capabilities has been treated as vendor-specific and not defined (e.g., the 3G cellular IMT-2000 standards); 2) a protocol is defined to determine which of the available capabilities in the T or R should be employed in a specific situation. As example, telephone modems prior to V.32 (circa 1984) selected the modulation used based on convention and vendor-specific decision boundaries. After V.32, the identification, negotiation and selection of a specific modulation was defined by an independent protocol, V.8.

### B. Defining Adaptability

*Adaptability* in communications systems is the process of automatically negotiating possible parameters, as it makes a system more adaptable. As defined here, adaptability requires three specific functions: identification of the capabilities available at each end, negotiation to determine the desired state-pairs (the interface), and selection of the desired state-pairs (which may require accessing software from elsewhere). These three functions are more complex versions of the three basic operations (Table 1) required for any communications: select input, compare input to reference (with adaptability mechanisms each end is

 ISBN: 978-1-61208-109-0 

compared to the other), and create output (select state-pairs). After these adaptable processes are completed, then communications of information or control can begin via the negotiated interface.

Interfaces have three major variations: fixed (state-pairs are unchangeable), flexible (state-pairs may be changed) and adaptable (state-pairs are negotiated). A mechanical plug and socket is an example of a fixed interface. Examples of flexible interfaces include: an Edison light bulb socket which supports many different types of lamps. While the mechanical aspects of the light bulb and socket are fixed, the load can be changed. A human user manually identifies and selects the specific lamp and the Edison light bulb plug/socket (the physical interface) makes this flexibility possible. A protocol example of a fixed interface that supports flexibility is the use of the Internet protocols TCP/IP as the interface with which each lower physical network or higher layer protocol is designed to interoperate. Some programmable processes also provide flexible interfaces. As example, XML (eXtensible Markup Language) provides identification and selection without negotiation.

Flexible interfaces using XML are the current state-of-the-art. Universal Plug and Play (UPnP) utilizes XML to identify and select resources between a local personal computer and peripheral devices. In this system the personal computer is "in charge" and negotiation is not necessary. Application layer negotiation to purchase music, books or programs is also widely used (e.g., Apple Store). But the negotiation of lower layer communications services (e.g., bandwidth, compression, security) where there may be a charge involved requires that the communications system itself be adaptable. As example, the compression of audio and video may be quite complex and many implementations are patented. The ability to select which form of audio or video compression is desired depends on the bandwidth available, the users' needs for audio or video quality and the users' willingness to pay. This requires a negotiation only possible with an adaptable communications system. Since the concept of adaptability is not widely understood, there are few examples of lower layer communications systems that support negotiation for the purpose of economic transactions. Yet the difficulty of equitably providing additional Internet bandwidth is well known and the even the subject of US Federal Communications Commission hearings.

A negotiation process is required to equitably support charging for lower layer communications services on the Internet. If one or more parameters are identified as proprietary (e.g., identified by a trademark), the use of such parameters would legally require the trademark owner's approval. All the other parameters used in the interface remain in the public domain. Such approval might require some form of payment to the trademark owner. If the proprietary service is valuable, implementers or users will have reason to pay the trademark owner for the use of their proprietary technology or service. Many different procedures are possible to compensate the trademark owner: charge for downloads, per implementation fees, usage fees, periodic maintenance/support fees, or simply the sales advantages of proprietary implementations or services offering improved operation over public sections of the standard.

Adaptability, which includes the means to negotiate state-pairs, is necessary to support true peer-to-peer communications. Without negotiation, one E/P must depend on the other to determine state-pairs, when unpaired states are possible. By definition, a dependent relationship cannot be peer-to-peer. Only when the two communicating E/P can change independently can they be considered peers.

### C. Measuring Adaptability

Communications interfaces are layered. Adaptability or flexibility may be employed at each layer of the OSI model or partially in one or more layers. A complex interface may consists of a mix of fixed, flexible and adaptable parameters. Understanding state-pairs and the significance of the number of state-pairs allows the interface's adaptability to be measured. The adaptability of an interface may be quantified by counting the total number of state-pairs available across the interface and relating it to the total number of flexible (e.g., weighted 0.5) and adaptable (e.g., weighted 1.0) state-pairs. Arbitrarily weighting flexibility as one-half adaptability allows both approaches to changing state-pairs to be evaluated. The choice of the weighting of flexibility and adaptability as well as the weighting of specific parameters (e.g., is physical layer adaptability more or less important than higher layer adaptability) is closely related to the intended use. Equation (6) models the example of a very simple interface with three parameters b, c and d where b has n state-pairs (all fixed), c has m state-pairs (all flexible) and d has p state-pairs (all adaptable); the percentage of adaptability (A) is shown in (7).

$$a = ( .5m+1p)/(n+m+p) \quad (6)$$

$$a \times 100 = A \text{ in percent} \quad (7)$$

Consider the example of the meter scale (n = 1000mm) used to transfer a measurement from one entity to another. When a meter scale is used on one side of a measurement stick and an extended yard scale (n = 16 x 39.37 inches = 629.92) is used on the other side, flexibility is introduced. Then each scale of the measuring stick is a set of fixed state pairs and the two scales are two flexible state-pairs (m=2):

1. measuring both entities with the meter scale.
2. measuring both entities with the yard scale.

The measurement of adaptability for this example is shown in (8), (9) and (10), where m=2, p=0, $n_1$=1000 and $n_2$=629.92:

$$a = ( .5x2+1x0)/(1000 + 629.92+2+0) \quad (8)$$

$$a = ( 1)/(1631.92) \quad (9)$$

$$a \times 100 = .06\% A \quad (10)$$

### D. Implementing Adaptability

At least two means to implement adaptability are known. Adaptability may be created by a software program (often termed agent software) that can identify, negotiate and select the state-pairs across an interface. Or an independent communications protocol may be used for the purposes of identification, negotiation and selection. When such a protocol is used only for these purposes, it is termed an *etiquette* [7]. It seems likely that other approaches to implement adaptability will be identified.

Etiquettes are already used in some communications systems, e.g., ITU V.8 for telephone modems, ITU T.30 for G3 fax, ITU G.994.1 for digital subscriber line transceivers, and IETF Session Initiation Protocol (SIP); their properties have been explored previously [7]. In the long term evolution (LTE) architecture, an etiquette would allow the service provider to negotiate the protocol that optimizes system loading or maximizes geographic coverage, or allow a user to select the protocol (and related service provider) that offers the best economic performance for that user. An etiquette also better supports troubleshooting of incompatibilities as each end can identify the available parameter sets of the other end. The use of adaptability mechanisms is a system architecture choice which significantly enhances the long term performance of programmable heterogeneous communications systems.

When systems are programmable, adaptability is possible. An etiquette transmitter presents the range of possible compatible parameters to an etiquette receiver. The etiquette receiver responds with its range of possible compatible parameters. Using heuristics local to the transmitter and receiver (e.g., largest parameter is best [pels, bits, colors, data rate, etc.]) or remote heuristics accessed by both the transmitter and the receiver (e.g., using a remote data base to determine which common parameters are to be utilized), the etiquette transmitter and receiver negotiate and select the desired interface for compatibility and follow-on communications.

Adaptability could be useful in software defined radios. A software defined radio which includes the physical layer, perhaps others, is not defined as adaptable but has the properties ─ programmable and a radio interface (non-mechanical) ─ that allow it to be adaptable.

Compatible systems have state-pairs. If there are transmitter states (at any OSI layer) that do not have related receiver states, such inconsistencies can cause "bugs." Adaptability mechanisms offer a means to negotiate and select a specific interface and reduce such bugs.

For the negotiation process of an etiquette to operate consistently, any addition to an etiquette must be a proper super-set of the previous version. As long as the etiquette is a logical single tree structure, where each branch refers to a single parameter set, no deletions are allowed and the etiquette receiver ignores anything it doesn't recognize, a correctly modified etiquette will always be backward compatible.

Following this model, an etiquette may be expanded whenever desired, independently in the transmitter and the receiver. This allows new capabilities, and the parameters in the etiquette that identify them, to be added to a communications system at any time. If both ends can support the new parameters they can be employed. If one end supports a parameter and the other end does not, it may be practical for the deficient end to download the needed software from a known Internet web site.

## VIII. CONCLUSIONS

Adaptability makes it possible to automatically negotiate the rising complexity of communications, introduce new technology into communications channels at will, simplify communications troubleshooting, better support multi-mode operation, avoid identified communications channel bugs and support incentives to developers and implementers without forcing all users of public interfaces to pay private fees. The advantages of adaptability are significant. Using the approach outlined, the adaptability of current and future programmable communications systems may be measured so that users, service providers and developers may easily recognize and utilize this important functionality.

## REFERENCES

[1] C. E. Shannon and W. Weaver, *The Mathematical Theory of Communications*, Fig. 1 p. 34. Urbana and Chicago IL, USA: University of Illinois Press, 1963.

[2] B. Russell, *Introduction to Mathematical Philosophy*, page 15. New York: Simon and Schuster, 1971.

[3] I. Kant, *Logic* (General Doctrine of Elements, Para. 6, Logical Acts of Comparison, Reflection and Abstraction), Library of Liberal Arts, trans. R.S. Hartman and W. Schwarz. Indianapolis and New York: The Bobbs-Merrill Company, Inc., 1974.

[4] N. Campbell, *Foundations of Science*, p. 267, Dover Publications, New York, NY, 1957.

[5] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, New York: John Wiley & Sons, Inc., 1991.

[6] H. Everett, III, Theory of the Universal Wave Function, page 25, 1957, published in *The Many-Worlds Interpretation of Quantum Mechanics*, edited by Bryce S. DeWitt and Neil Graham, Princeton Series in Physics, 1980.

[7] K. Krechmer, "Fundamental nature of standards: technical perspective," *IEEE Communications Magazine*, 38(6), p. 70, June, 2000. Also at http://www.csrstds.com/fundtec.html

 ISBN: 978-1-61208-109-0 

# Adaptability Support in Time- and Space-Partitioned Aerospace Systems

João Craveiro and José Rufino
*Universidade de Lisboa, Faculdade de Ciências, LaSIGE*
*FCUL, Ed. C6, Piso 3, Campo Grande, 1749-016 Lisbon, Portugal*
*E-mail: jcraveiro@lasige.di.fc.ul.pt, ruf@di.fc.ul.pt*

***Abstract*—The AIR (ARINC 653 in Space Real-Time Operating System) technology targets modern aerospace systems, where the concepts of time- and space-partitioning are applied. AIR features advanced timeliness control and adaptation mechanisms in its design, such as mode-based schedules, process deadline violation monitoring, and protection against event overload. The timing parameters of a space mission may vary throughout time, according to its mode/phase of operation, and the spacecraft may be exposed to unpredictable events and failures. In this paper we explore the adaptation potential of the advanced features included in AIR, analysing their code complexity (which influences software verification, validation and certification efforts) and computational complexity (which correlates to the temporal overhead impact on the system), and discussing how they can be applied to provide more adaptive, reconfigurable and self-adaptive AIR-based systems.**

***Keywords*-adaptive systems; aerospace industry; logic partitioning; processor scheduling; real-time systems**

## I. INTRODUCTION

The computing infrastructure aboard spacecrafts employs embedded systems to cope with strict dependability and real-time requirements. Additionally, cost concerns call for flexible resource reallocation and the overall reduction of size, weight and power consumption (SWaP). To address SWaP requisites, functions which traditionally received dedicated resources are now integrated in a shared computing platform. As these functions may have different degrees of criticality and predictability and/or originate from multiple suppliers, this integration brings potential safety hazards, for which the architectural approach of time- and space-partitioning (TSP) has been proposed. Applications are separated into logical containers, *partitions*, for the benefit of fault containment, software integration, and independent verification, validation and certification processes. The aviation industry had already made a similar move, by transitioning from federated architectures to Integrated Modular Avionics (IMA) [1].

The AIR architecture [2] was defined in response to the interest of space agencies, namely the European Space Agency (ESA), and industry partners in applying TSP concepts to the spacecraft onboard computing resources [3]. Each partition hosts its own application and operating system, either a real-time operating system (RTOS) or a generic non-real-time one. AIR employs a two-level scheduling scheme. Partitions are scheduled under a predetermined, cyclically repeated, sequence of time windows. Inside each partition's time windows, the respective processes compete according to the native operating system's process scheduler policy.

In normal conditions, a mission goes through multiple phases (flight, approach, exploration) [4]. Additionally, unplanned circumstances may happen, such as unforeseeable external events and internal failures. Adaptation to changing/unexpected conditions is of great importance for a mission's survival. This kind of adaptation can include mechanisms of support for assisted adaptation, reconfiguration capabilities, and self-adaptation according to environmental information. Such need is more stringent in the case of unmanned missions. Flexible adaptation to unforeseen events is of paramount importance, and has been proven to prolong the lifetime of unmanned space vehicles by years [5].

In this paper, we explore the adaptation, reconfiguration and self-adaptation potential of the AIR architecture, with respect to timeliness control and failure detection/recovery.

### *Related work*

To the best of our knowledge, the only contemporary approach to flexible scheduling in a TSP system is the mode-based scheduling feature provided by the commercial Wind River VxWorks 653 solution [6]. Previous academic research on TSP solutions [7] and works on scheduling analysis for TSP systems [8], [9], [10] do not state including or foreseeing mechanisms for timing parameters adaptation. Architectural alternatives to TSP/IMA are compared in [11] in terms of features which contribute to adaptability, and recommendations are made to include adaptive features in IMA-like architectures. Preliminary results are presented in [12] for a reconfigurable IMA platform. Emergence and increasing acceptance of adaptive and reconfigurable control in unmanned aerial systems is pointed out in [13].

### *Paper outline*

Section II provides an overview of the AIR architecture for self-containment of the issues at hand. Section III describes the advanced timeliness control mechanisms provided by AIR, while Section IV discusses how they allow adaptive behaviour. Section V exposes the implementation and evaluation of a prototype demonstrator of these capabilities. Finally, Section VI closes the paper with concluding remarks and future work directions.

 ISBN: 978-1-61208-109-0 

## II. AIR ARCHITECTURE FOR TSP SYSTEMS

The AIR design [2] evolved from a proof of feasibility for adding ARINC 653 support to the Real-Time Executive for Multiprocessor Systems (RTEMS) to a multi-OS (operating system) time- and space-partitioned (TSP) architecture. Its modular design aims at high levels of flexibility, hardware- and OS-independence, and independent component verification, validation and certification.

Each partition can host a different OS (the partition operating system, POS), which in turn can be either a real-time operating system (RTOS) or a generic non-real-time one. We will now describe the AIR architecture in enough detail for the scope of this paper. A more in-depth description of AIR can be found in [2].

### A. System architecture

The modular design of the AIR architecture is pictured in Figure 1. The *AIR Partition Management Kernel (PMK)* is the basis of the Core Software Layer of an AIR-based system. The AIR PMK hosts crucial functionality such as partition scheduling and dispatching, low-level interrupt management, and interpartition communication support.

The *AIR POS Adaptation Layer (PAL)* encapsulates the POS of each partition, providing an adequate POS-independent interface to the surrounding components.

The *APEX Interface* component provides a standard programming interface derived from the ARINC 653 specification [14], with the possibility of being subsetted and/or adding specific extensions for certain partitions [15].

AIR also incorporates *Health Monitoring (HM)* functions to contain faults within their domains of occurrence and to provide the corresponding error handling capabilities. Support to these functions is spread throughout virtually all of the AIR architecture's components.

### B. Two-level scheduling scheme

The AIR technology employs a two-level scheduling scheme (Figure 2). The first level corresponds to partition scheduling and the second level to process scheduling. Partitions are scheduled on a cyclic basis, according to a partition scheduling table (PST) repeating over a major time frame (MTF). This table assigns execution time windows to

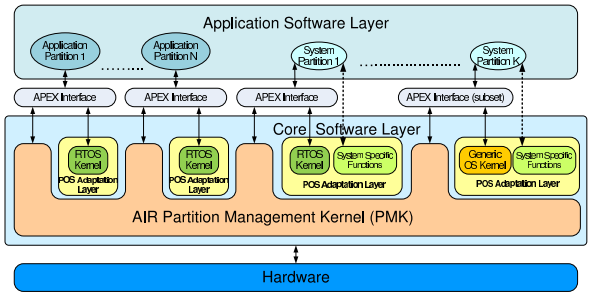



Figure 1. AIR system architecture

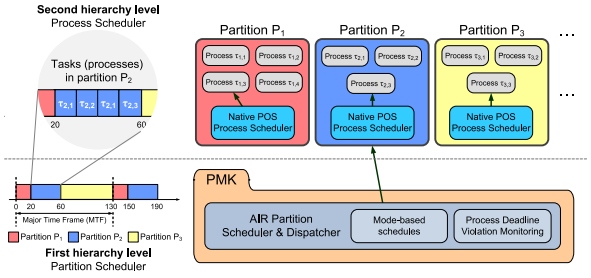



Figure 2. Two-level scheduling scheme

partitions. Inside each partition's time windows, its processes compete according to the POS's native process scheduler.

## III. DESIGNING FOR ADAPTABILITY

The potential to adapt to changing environmental or operating conditions is crucial for space missions. Existing studies show that a number of failures may be mitigated by software reconfigurability [5].

AIR employs special design and engineering decisions to address specific adaptation requirements, namely on the temporal domain and on failure detection and recovery.

### A. Mode-based schedules

Timing requirements are among the conditions which may change according to a mission's phase, since certain functions should only execute during certain phases. Under the basic mandatory scheme defined in ARINC 653 [14], there is a single fixed PST under which all partitions are scheduled. This PST must be tailored to attended to the temporal requisites of the partitions in all of the phases the mission goes through. Generating such a PST is a difficult task (even assisted by a tool), and the end result may be a mission with frequent resource utilization waste.

AIR approaches this issue with the notion of *mode-based schedules*, inspired by the optional service defined in ARINC 653 Part 2 [16]. Instead of one fixed PST, the system can be configured with multiple PSTs, which may differ in terms of the MTF duration, of which partitions are scheduled, and of how much processor time is assigned to them. The system can then switch between these PSTs; this is performed through a service call issued by an authorized and/or dedicated partition. To avoid violating temporal requirements, a PST switch request is only effectively granted at the end of the ongoing MTF.

The AIR Partition Scheduler, with the addition of mode-based schedules, functions as described in Algorithm 1. The first verification to be made is whether the current instant is a partition preemption point (line 2). In case it is not, the execution of the partition scheduler is over; this is both the best case and the most frequent one. If it is a partition preemption point, a verification is made (line 3) as to whether there is a pending scheduling switch to be applied and the current instant is the end of the

 ISBN: 978-1-61208-109-0 

**Algorithm 1** AIR Partition Scheduler featuring mode-based schedules

1: $ticks \leftarrow ticks + 1$ ▷ $ticks$ is the global system clock tick counter
2: **if** $schedules_{currentSchedule}.table_{tableIterator}.tick = (ticks - lastScheduleSwitch) \bmod schedules_{currentSchedule}.mtf$
3: **if** $currentSchedule \neq nextSchedule \wedge (ticks - lastScheduleSwitch) \bmod schedules_{currentSchedule}.mtf = 0$
4: $currentSchedule \leftarrow nextSchedule$
5: $lastScheduleSwitch \leftarrow ticks$
6: $tableIterator \leftarrow 0$
7: **end if**
8: $heirPartition \leftarrow schedules_{currentSchedule}.table_{tableIterator}.partition$
9: $tableIterator \leftarrow (tableIterator + 1) \bmod schedules_{currentSchedule}.numberPartitionPreemptionPoints$
10: **end if**

MTF. If these conditions apply, then a different PST will be used henceforth (line 4). The partition which will hold the processing resources until the next preemption point, dubbed the heir partition, is obtained from the PST in use (line 8) and the AIR Partition Scheduler will now be set to expect the next partition preemption point (line 9). Generation of different partition schedules can be aided by a tool that applies rules and formulas to the temporal requirements of the constituent processes of the necessary partitions [2], [17].

### B. Process deadline violation monitoring

During system execution, it may be the case that a process exceeds its deadline. This can be caused by a malfunction, by transient overload (e. g., due to abnormally high event signalling rates), or by the underestimation of a process's worst case execution time (WCET) at system configuration and integration time. Factors related to faulty system planning (such as the time windows not satisfying the partitions' timing requirements) could, in principle, also cause deadline violations. However, such issues can be predicted and avoided using offline tools that verify the fulfilment of timing requirements [2], [17].

In addition, it is also possible that a process exceeds a deadline while the partition in which it executes is inactive. This violation will only be detected when the partition is being dispatched, just before invoking the process scheduler.

Deadline verification, which is invoked for the currently active partition immediately following the announcement of elapsed system clock ticks, obeys to Algorithm 2. In the absence of a violation, only the earliest deadline is checked. Subsequent deadlines may be verified until one has not been missed. This methodology is optimal with respect to deadline violation detection latency, which upper bound for each partition is the maximum interval between two consecutive time windows.

Figure 3 provides a use-case scenario for the process deadline violation monitoring functionality. The exemplified process (with the identifier $pid$) has a relative deadline of $t_3 - t_1$. When it becomes ready at $t = t_1$ via the START primitive, its absolute deadline time is set to $t_3$. At $t = t_2$,

**Algorithm 2** Deadline verification at the AIR PAL level

1: *PAL_CLOCKTICKANNOUNCE($elapsedTicks$)
2: **for all** $d \in PAL\_deadlines$ **do**
3: **if** $d.deadlineTime \geq$PAL_GETCURRENTTIME( )
4: **break**
5: **end if**
6: HM_DEADLINEVIOLATED($d.pid$) ▷ $pid$: process identifier
7: PAL_REMOVEPROCESSDEADLINE($d$)
8: **end for**

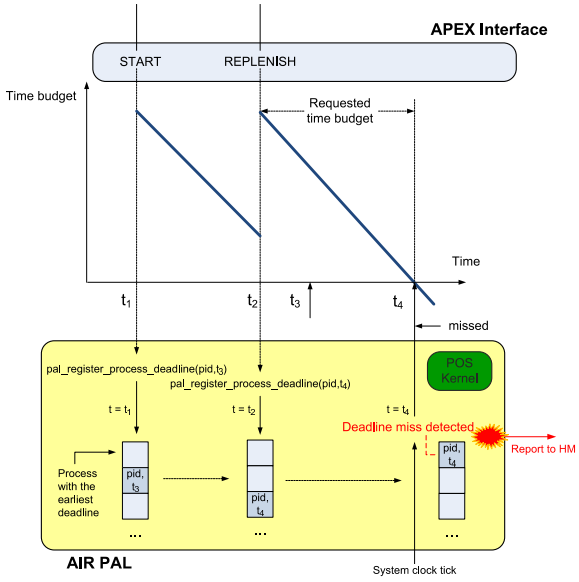



Figure 3. Process deadline violation monitoring example

the REPLENISH primitive is called, requesting a deadline time replenishment, so that the new absolute deadline time is $t_4$. When the instant $t = t_4$ arrives and the process $pid$ has not yet finished its execution (e. g., having called the PERIODIC_WAIT primitive to suspend itself until its next release point), its deadline will be the earliest registered in the respective AIR PAL data structure. The violation is then detected, and reported to the Health Monitor.

### C. Health Monitor and error handling

In the context of fault detection and isolation, ARINC 653 classifies process deadline violation as a process level error (that impacts one or more processes in the partition, or the entire partition) [14]. The action to be performed in the event of an error is defined by the application programmer through an appropriate error handler [2].

The error handler is an application process tailored for partition-wide error processing. The occurrence of process-/partition-level errors may be signalled through interpartition communication to a (system partition) process performing a Fault Detection, Isolation and Recovery (FDIR) function for the spacecraft [4]. A system-wide reconfigurability logic should be included in FDIR.

 ISBN: 978-1-61208-109-0 

Table I
ESSENTIAL APEX SERVICES TO SUPPORT MODE-BASED SCHEDULES

| Primitive | Short description |
|---|---|
| SET_SCHEDULE | Request for a new PST to be adopted at the end of the current MTF |
| GET_SCHEDULE_STATUS | Obtain current schedule, next schedule (same as current schedule if no PST change is pending), and time of the last schedule switch |

Table II
APEX SERVICES IN NEED OF MODIFICATIONS TO SUPPORT PROCESS DEADLINE VIOLATION MONITORING

| Primitive | Short description |
|---|---|
| **Need to register/update deadline** | |
| [DELAYED_]START | Start a process [with a given delay] |
| PERIODIC_WAIT | Suspend execution of a (periodic) process until the next release point |
| REPLENISH | Postpone a process's deadline time |
| **Need to unregister deadline** | |
| STOP[_SELF] | Stop a process [itself] |

Table III
ESSENTIAL APEX SERVICES FOR HEALTH MONITORING

| Primitive | Short description |
|---|---|
| RAISE_APPLICATION_ERROR | Notify the error handler for a specific error type |

### D. Impact on APEX services

To allow application programmers to use the mode-based schedules functionality, the APEX interface is extended with additional primitives presented in Table I.

Process deadline monitoring calls for adaptation of process management services, encapsulated by appropriated AIR PAL functions. Table II shows the APEX primitives which need to register deadline information (either updating the deadline information for the process, or inserting it if it does not exist yet) or unregister deadline information (removing any information on the respective process from the AIR PAL deadline verification data structures). The APEX primitive RAISE_APPLICATION_ERROR, described in Table III, is used to report a process deadline violation to the HM and trigger the defined error handler.

### E. Low-level event overload control

The AIR PMK includes advanced adaptation mechanisms to control the timeliness of asynchronous events signalled through processor interrupts. These mechanisms use the discrete event processing methodology described in [18], and have been adapted for application on the AIR architecture. The resulting approach, integrating with the Health Monitor, is presented in Algorithm 3. The continuous time, $t$, is transformed into a discrete time $n$ through a *sample/hold* function (line 2). The discrete time is used for the determination of the event (interrupt) metrics, such as its rate of occurrence, using recursive digital filters (line 3). The complexity of computing some filtering functions inside the interrupt service routing (ISR) is removed resorting to previously built lookup tables [18]. If a specific metric upper threshold $M$ is exceeded, interrupt service is disabled and the HM is notified to take action. The corresponding error handler then polls events at a predefined fixed rate. The effectiveness of this approach is illustrated in Figure 4, using an Infinite Impulse Response (IIR) filter for rate determination [18]. When the overload state is over, and the relevant metric value $y_n$ decreases below the lower threshold $m$, the error handler enables the interrupts and returns to the state of waiting for further overload notifications.

**Algorithm 3** Event (interrupt) overload control

```
1: t ← GETTIME()
2: n ← SAMPLEHOLD(t)
3: y_{n+1} ← FILTERING(n, n_last, y_n)        ▷ Event metric determination
4: n_last ← n
5: if y_n > M                                  ▷ Overload detection
6:   DISABLEINTERRUPT(irq)                     ▷ irq: interrupt request number
7:   HM_EVENTOVERLOAD(irq)
8: end if
```

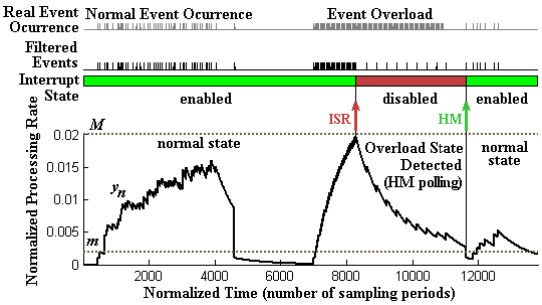



Figure 4. Operation of overload control mechanisms [18]

## IV. ACHIEVING ADAPTABILITY

The described features can be used to allow the adaptation of the system to changing conditions, either by the action of a human operator (adaptability) or autonomously after processing acquired information (self-adaptability).

### A. Adaptability and reconfigurability

By offering the possibility to host multiple PSTs and switch among them by demand during the execution of the system, AIR allows for guided adaptation of the system to the mission's different phases and to detected operational condition changes.

The proposed reconfigurability model also allows updating the set of available PSTs during the operation of the mission, by issuing an update from ground control. This process can be extended to allow updating the applications running inside the partitions [15]. Process deadline violation monitoring information can be useful in detecting the need for PSTs and/or applications to be updated.

 ISBN: 978-1-61208-109-0 

### B. Self-adaptability

Besides ground control commands, a request to use a different PST can also be autonomously issued from the interpretation of the internal and external operating conditions of the mission. The Attitude and Orbit Control Subsystem (AOCS) [4] should detect when the mission should transit from flight to approach and from approach to exploration, for instance, and (propose to) switch schedules accordingly.

Another self-adaptation use of mode-based schedules can profit from process deadline violation monitoring. The system integrator can include several PSTs that fulfil the same set of temporal requirements for the partitions, but distribute additional spare time inside the MTF among these partitions in different ways. This set of PSTs, coupled with an appropriate HM handler for the event of process deadline violation, can be used to temporarily or permanently assign additional processing time to a partition hosting an application which has repeatedly observed to be missing deadlines.

An additional degree of self-adaptability is obtained by applying event overload control mechanisms with reconfigurable parameters to preserve timeliness [18].

## V. Proof-of-concept prototype and evaluation

The core mechanisms for adaptability and reconfigurability of the AIR architecture have been prototyped on both Intel IA-32 and SPARC LEON platforms. Each partition executes an RTEMS-based [19] mockup application representative of typical functions present in a spacecraft.

### A. Code complexity

Critical software, namely that developed to go aboard a space vehicle, goes through a strict process of verification, validation and certification. Code complexity increases the effort of this process and the probability of there being bugs.

One metric for code complexity is its size, in lines of source code. Since equivalent code can be arranged in ways which account for different lines of code counts, standardized accounting methods must be used. We employ the *logical source lines of code (logical SLOC)* metric of the Unified CodeCount tool [20]. Another useful metric is *cyclomatic complexity*, which gives an upper bound for the number of test cases needed for full branch coverage and a lower bound for those needed for full path coverage.

The C implementation of Algorithm 1 is accounted for in Table IV, which shows its logical SLOC count and cyclomatic complexity. Code introduced at the AIR PAL level to achieve process deadline violation monitoring is accounted for in Table V. The total complexity added in terms of code executed during a clock tick ISR is a small fraction of that already present in the underlying ISR code.

### B. Computational complexity analysis

Since the verifications of deadlines and of the need to apply a new PST are executed inside the system clock tick ISR, they must have a bounded execution time; computational complexity is a good indicator thereof.

In the AIR Partition Scheduler (Algorithm 1), all instructions are $\mathcal{O}(1)$. Accesses to multielement structures are made by index, and thus their complexity does not depend on the number of elements or the position of the desired element.

Linear complexity is easily achievable for the majority of the executions of the process deadline verification at the AIR PAL level (Algorithm 2). By placing deadlines in a linked list in ascending order of deadline times, the earliest deadline is retrieved in constant time. The removal of a violated deadline from the data structure can also be made in constant time, since we already hold a pointer for the said deadline. In case of deadline violation, the next deadline(s) will successively be verified until reaching one that has not expired. Thus, the worst case wields $\mathcal{O}(n)$, where $n$ is the number of processes in the partition. However, by design, the number of processes is bounded, and these worst cases are highly exceptional and mean the total and complete failure of the partition, which should be signalled to the HM.

Table IV
Logical SLOC and cyclomatic complexity (CC) for the AIR Partition Scheduler with mode-based schedules

| | Logical SLOC | CC |
|---|---|---|
| **AIR Partition Scheduler** [a] | 13 | 4 |
| **Underlying clock tick ISR** | >190 [b] | >20 |

[a] Algorithm 1
[b] C code only; plus >182 assembly instructions

Table V
Logical SLOC and cyclomatic complexity (CC) for the implementation of deadline violation monitoring in AIR PAL

| | Logical SLOC | CC |
|---|---|---|
| **Register deadline** | 34 | 6 |
| **Unregister deadline** | 12 | 3 |
| **Verify deadlines** [a] | 16 | 2 |

[a] Algorithm 2

Table VI
AIR Partition Scheduler (with mode-based schedules) execution time — basic metrics

| Minimum (ns) | Maximum (ns) | Average (ns) |
|---|---|---|
| 32 | 186 | 36 |

### C. Basic execution metrics

AIR Partition Scheduler execution time has been measured, resulting in the basic metrics shown in Table VI. These values were obtained executing the AIR prototype demonstrator on a native machine equipped with an Intel IA-32 CPU with a clock of 2833 MHz. Time readings are obtained from the CPU Time Stamp Counter (TSC)

 ISBN: 978-1-61208-109-0 

64-bit register, being rounded to 1 ns. Measurements showed negligible variation between sample executions.

Though encouraging, these results should be enriched. Further work will compare them with the execution of the whole clock tick ISR, dissect the execution time by subcomponents, and identify trends in execution time variation.

## VI. Conclusion and future work

In this paper, we detailed and analysed fundamental mechanisms providing support for adaptive and self-adaptive behaviour to aerospace missions based on the AIR architecture for time- and space-partitioned systems. Support for mode-based partition schedules directly provides the capacity for adaptation by direct order from ground control, and also for self-adaptation according to the perceived operational conditions of the mission. The mechanisms for process deadline violation monitoring and protection against event overload allow controlling the overall timeliness, through self-adaptation in the presence of exceptional and uncertain operational conditions.

Besides implementing self-adaptation as a function of process deadline violations, future work includes adding AIR the capability to receive and apply updates to the currently installed applications and PSTs [15]. We are also currently exploring the approach of taking advantage of multicore platforms in AIR [2]. The availability of multiple cores can be applied for fault tolerance, with the system adapting to a hardware failure by assigning an application to a different core than the (failed) current one.

## Acknowledgment

This work was partially developed within the scope of the European Space Agency Innovation Triangle Initiative program, through ESTEC Contract 21217/07/NL/CB, Project AIR-II (ARINC 653 in Space RTOS – Industrial Initiative). This work was partially supported by FCT (Fundação para a Ciência e a Tecnologia), through the Multiannual Funding and CMU-Portugal Programs and the Individual Doctoral Grant SFRH/BD/60193/2009.

## References

[1] AEEC, "Design guidance for Integrated Modular Avionics," Aeronautical Radio, Inc., ARINC Report 651-1, Nov. 1997.

[2] J. Rufino, J. Craveiro, and P. Verissimo, "Architecting robustness and timeliness in a new generation of aerospace systems," in *Architecting Dependable Systems 7*, ser. LNCS, A. Casimiro, R. de Lemos, and C. Gacek, Eds. Springer, 2010, accepted for publication.

[3] TSP Working Group, "Avionics time and space partitioning user needs," ESA, European Space Research and Technology Centre, Technical Note TEC-SW/09-247/JW, Aug. 2009.

[4] P. W. Fortescue, J. P. W. Stark, and G. Swinerd, Eds., *Spacecraft Systems Engineering, 3rd edition*. Wiley, 2003.

[5] M. Tafazoli, "A study of on-orbit spacecraft failures," *Acta Astronautica*, vol. 64, no. 2–3, pp. 195–205, Jan.–Feb. 2009.

[6] Wind River, "Wind River VxWorks 653 Platform 2.3," Apr. 2010, retrieved Jun 17, 2010. [Online]. Available: http://www.windriver.com/products/product-notes/PN_VE_653_Platform2_3_0410.pdf

[7] A. Crespo, I. Ripoll, and M. Masmano, "Partitioned embedded architecture based on hypervisor: the XtratuM approach," in *Proc. Eighth European Dependable Computing Conf.*, Valencia, Spain, Apr. 2010.

[8] N. Audsley and A. Wellings, "Analysing APEX applications," in *Proc. 17th IEEE Real-Time Systems Symp.*, Washington DC, USA, Dec. 1996, pp. 39–44.

[9] Y. Lee, D. Kim, M. Younis, and J. Zhou, "Partition scheduling in APEX runtime environment for embedded avionics software," in *Proc. 5th Int. Conf. on Real-Time Computing Systems and Applications*, Hiroshima, Japan, 1998.

[10] A. Easwaran, I. Lee, O. Sokolsky, and S. Vestal, "A compositional scheduling framework for digital avionics systems," in *Proc. 15th IEEE Int. Conf. Embedded Real-Time Computing Systems and Applications*, Beijing, China, Aug. 2009.

[11] B. Ford, P. Bull, A. Grigg, L. Guan, and I. Phillips, "Adaptive architectures for future highly dependable, real-time systems," in *Proc. 7th Ann. Conf. on Systems Engineering Research*, Loughborough, United Kingdom, Apr. 2009.

[12] P. Bieber, E. Noulard, C. Pagetti, T. Planche, and F. Vialard, "Preliminary design of future reconfigurable IMA platforms," in *Second Int. Workshop on Adaptive and Reconfigurable Embedded Systems*, Grenoble, France, Oct. 2009, pp. 21–24.

[13] B. Vanek, "Future trends in UAS avionics," in *Proc. 10th Int. Symp. of Hungarian Researchers on Computational Intelligence and Informatics*, Budapest, Hungary, Nov. 2009.

[14] AEEC, "Avionics application software standard interface, part 1 - required services," Aeronautical Radio, Inc., ARINC Specification 653P1-2, Mar. 2006.

[15] J. Rosa, J. Craveiro, and J. Rufino, "Exploiting AIR composability towards spacecraft onboard software update," in *Actas do INForum - Simpósio de Informática 2010*, Braga, Portugal, Sep. 2010.

[16] AEEC, "Avionics application software standard interface, part 2 - extended services," Aeronautical Radio, Inc., ARINC Specification 653P2-1, Dec. 2008.

[17] J. Craveiro and J. Rufino, "Schedulability analysis in partitioned systems for aerospace avionics," in *Proc. 15th IEEE Int. Conf. on Emerging Technologies and Factory Automation*, Bilbao, Spain, Sep. 2010.

[18] M. Coutinho, J. Rufino, and C. Almeida, "Control of event handling timeliness in RTEMS," in *Proc. 17th IASTED Int. Conf. on Parallel and Distributed Computing and Systems*, Phoenix, USA, Nov. 2005.

[19] *RTEMS C Users Guide*, 4.8 ed., OAR Corp., Feb. 2008.

[20] V. Nguyen, S. Deeds-Rubin, and T. Tan, "A SLOC counting standard," in *The 22nd Int. Ann. Forum on COCOMO and Systems/Software Cost Modeling*, Los Angeles, USA, 2007.

 ISBN: 978-1-61208-109-0 

# An Adaptive Look-Ahead Strategy-Based Algorithm for the Circular Open Dimension Problem

Hakim Akeb
*ISC Paris School of Management*
*22 Boulevard du Fort de Vaux*
*75017 Paris, France*
*UPJV, UR MIS, Équipe ROAD*
*Email: hakeb@groupeisc.com*

Mhand Hifi
*Université de Picardie Jules Verne*
*UR MIS, Équipe ROAD*
*33 rue Saint-Leu, Amiens, France*
*Email: mhand.hifi@u-picardie.fr*

***Abstract*—In this paper, we study the *circular open dimension problem*, a well-known combinatorial optimization problem of the cutting and packing family. We are given a set of circular pieces (or circles) of known radii and a strip of fixed width and unlimited length. The objective is to determine the minimum length of the initial strip that packs all the circular pieces. The problem is approximately solved with an adaptive look-ahead strategy-based algorithm, which combines greedy procedures, restarting and separate beams strategies, and a look-ahead search. The running experiments show, on a set of benchmark instances of the literature, the effectiveness of the proposed method. For these instances, the proposed algorithm improves 10 results out of 18.**

***Keywords*-beam search, cutting and packing, look-ahead, minimum local-distance position, multi-start strategy.**

## I. INTRODUCTION

Some industrial processes need to pack efficiently goods or products in order to save space during the storage or the transportation phase. In other cases, industrial machines have to cut material of predetermined shapes from a given two-dimensional plate. *Cutting and Packing* (C&P) problems have several industrial and commercial applications and they were intensively studied by applying approximate and exact algorithms (cf. Wäscher *et al.* [1]). In fact, a C&P problem consists of cutting or packing a set of items of known dimensions from or into one or more large objects or containers so as to minimize the unused portion of the objects, or waste. The items and objects can have rectangular, circular, or irregular shapes. These problems are known as difficult to solve exactly and so, heuristics are used for tackling a variety of them. Herein, we solve the *circular open dimension problem* (CODP) where the items are circular and the container is a strip. CODP is also known as the strip cutting/packing problem (cf. Akeb and Hifi [2], Hifi and M'Hallah [3], and Huang *et al.* [4]).

More precisely, in CODP we are given an initial strip $S$ of fixed width $W$ and unlimited length ($L$), as well as a finite set $N$ of $n$ circular pieces $C_i$ of known radius $r_i,\ i \in N = \{1, \ldots, n\}$. The objective is to pack (or cut) all the pieces such that (i) the length of the strip $S$ is minimized and (ii) there is no overlapping between pieces, and between pieces and the edges of the strip.

CODP can be formulated as follows:

$$\begin{cases} \min L & & \\ (x_i - x_j)^2 + (y_i - y_j)^2 & \geq (r_i + r_j)^2,\ j < i,\ (i,j) \in N^2 & (1) \\ x_i - r_i & \geq 0,\ \forall i \in N & (2) \\ y_i - r_i & \geq 0,\ \forall i \in N & (3) \\ W - y_i - r_i & \geq 0,\ \forall i \in N & (4) \\ L - x_i - r_i & \geq 0,\ \forall i \in N & (5) \\ L & \geq \underline{L} & (6) \end{cases}$$

Equation (1) represents the non-overlap constraint of any pair of distinct pieces $(C_i, C_j)$; it means that the distance between the centers of these two circles must be greater than or equal to the sum of their radii. Equations (2), (3), (4), and (5) ensure that any piece $C_i,\ i \in N$, belongs to the target rectangle of dimensions $(L, W)$. Finally, Equation (6) means that the solution $(L)$ is necessarily greater than or equal to the trivial lower bound $\underline{L} = \left(\pi \times \sum_{i \in N} r_i^2\right)/W$.

The rest of the paper is organized as follows. Section II provides a literature review for CODP. Section III summarizes the principle of the minimum local-distance position procedure. Section IV describes the BSBIS algorithm already considered in Akeb and Hifi [2]. Section V details the proposed adaptive look-ahead strategy-based algorithm, which combines beam search (using separate beams), a multi-start, and a look-ahead strategy. In Section VI, the results of the proposed algorithm are evaluated on a set of benchmark instances. Finally, Section VII summarizes the contribution of this work and indicates some perspectives.

## II. LITERATURE REVIEW

Several approaches and methods were used in order to solve circle cutting/packing problems. These approaches depend generally on whether the container is a circle or a rectangle.

The problem of packing identical circles into a circular container was for example studied by Graham *et al.* [5]. Their approach is based on combining both billiard simulation and energy function minimization techniques. The problem of packing non-identical circles of known radii into the smallest containing circle was studied by several authors. Indeed, Huang *et al.* [6] have proposed two algorithms (called A1.0 and A1.5), which are based on the maximum hole degree

 ISBN: 978-1-61208-109-0 

(MHD) strategy. Hifi and M'Hallah [7] proposed a dynamic adaptive local search where, at each iteration, a new circle is inserted, and the coordinates of the container and its radius are updated. Finally, Akeb *et al.* [8] applied, for the same problem, a width-first beam search.

For the rectangular (or strip) version of the problem, George *et al.* [9] designed an approach that simulates the packing of circles into a rectangle. The authors showed that the best rules are those using a quasi-random approach and a genetic algorithm. Stoyan and Yaskov [10] solved the CODP by considering a mathematical model that searches for feasible local optima by combining a tree-search procedure with a reduced-gradient method. Hifi and M'Hallah [11] proposed a constructive procedure and a genetic algorithm to pack circles into a strip. Birgin *et al.* [12] tackled the same problem by proposing an algorithm based on a non-linear approach. Huang *et al.* [4] proposed two solution procedures for the CODP and finally, Akeb and Hifi [2] proposed three algorithms for the latest problem: (i) an open-strip generation solution procedure based on an optimization problem, (ii) a local beam-search solution procedure that combines beam search and the open-strip generation procedure, and (iii) a hybrid algorithm that combines beam search, binary interval search, and the open-strip generation procedure.

In this paper, CODP is solved by using an adaptive algorithm. The method combines separate beams search, a multi-start strategy, and look-ahead. So the objective is to see how the look-ahead may improve the existing results obtained by using the other strategies.

## III. THE MLDP STRATEGY

Herein, we describe the notations used in the paper and explains the principle of the minimum local distance position (MLDP) strategy.

**Notations.**

1) $N = \{1, ..., n\}$ denotes the set of the circles to pack,
2) the strip $S$ is placed with its bottom left corner at $(0, 0)$,
3) the four edges of $S$ are denoted by $S_{\text{left}}$, $S_{\text{top}}$, $S_{\text{right}}$, and $S_{\text{bottom}}$, respectively,
4) each circular piece $C_i$ of radius $r_i$ is placed with its center at $(x_i, y_i)$,
5) $I_i$ is the set of $i$ circles already packed inside the strip,
6) $\overline{I}_i$ contains the circles, which are not yet placed,
7) $P_{I_i}$ denotes the set of distinct corner positions of $C_{i+1}$ given the set $I_i$,
8) a corner position $p_{i+1} \in P_{I_i}$ is determined by using two elements $a$ and $b$. An element is either a piece already placed or one of the three edges of $S$ ($S_{\text{left}}$, $S_{\text{top}}$, $S_{\text{bottom}}$). $T_{p_{i+1}}$ denotes the set composed of both elements $a$ and $b$.

The *minimum local distance position* (MLDP) can be applied in a greedy way in order to select a corner position among a set of feasible positions. An iterative MLDP selection induces a greedy algorithm that can be summarized as follows. Having positioned a set $I_i \subset N$ of circular pieces, the process tries to position the next piece, namely $C_{i+1}, i \in N \backslash I_i$, into a corner position among all its eligible positions in the strip $S$, i.e., without overlapping any of the pieces already placed, and by touching two already placed circles or by touching a circle and one of the edges of the strip.

Fig. 1. Feasible distinct corner positions for $C_3$ in the strip

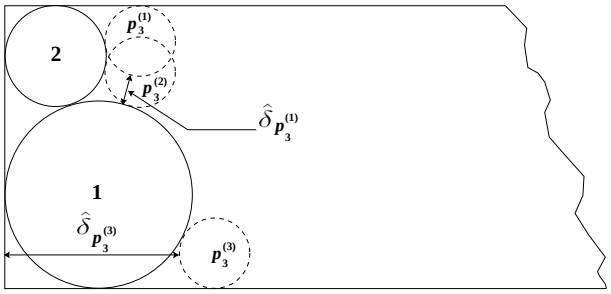



Figure 1 illustrates such corner positions −dotted-line circular pieces− for circle $C_3$. Note that $I_2 = \{C_1, C_2\}$ and $P_{I_2} = \{p_3^{(1)}, p_3^{(2)}, p_3^{(3)}\}$. The notation $p_3^{(k)}$, for $k = 1, \ldots, 3$, denotes the $k^{th}$ corner position such that $p_3^{(k)} \in I_2$. In this example, the corner position $p_3^{(1)}$ is obtained by using the piece $C_2$ and the top edge of the strip, $p_3^{(2)}$ is provided by using both pieces $C_1$ and $C_2$, and $p_3^{(3)}$ is obtained by using $C_1$ and the bottom edge of the strip. It follows that $T_{p_3^{(1)}} = \{C_2, S_{\text{top}}\}$, $T_{p_3^{(2)}} = \{C_1, C_2\}$ and $T_{p_3^{(3)}} = \{C_1, S_{\text{bottom}}\}$.

Let $C_{i+1}$ be the selected circular piece to pack at position $p_{i+1}$ and let $\delta_{i+1}(edge)$, $edge \in E_{\text{edge}} = \{S_{\text{left}}, S_{\text{bottom}}, S_{\text{top}}\}$, be the three distances defined as follows: $\delta_{i+1}(S_{\text{left}}) = x_i - r_i$, $\delta_{i+1}(S_{\text{bottom}}) = y_i - r_i$, and $\delta_{i+1}(S_{\text{top}}) = W - y_i - r_i$.

The distance between the edge of the next circle to place $C_{i+1}$ (when positioned at $p_{i+1}$) and circle $C_j$ is denoted by $\delta_{i+1}(j)$ and is defined as follows:

$$\delta_{i+1}(j) = \sqrt{(x_{i+1} - x_j)^2 + (y_{i+1} - y_j)^2} - (r_{i+1} + r_j). \quad (7)$$

It follows that the MLDP of a piece $C_{i+1}$ when positioned at $p_{i+1} \in P_{I_i}$ is:

$$\hat{\delta}_{p_{i+1}} = \min_{\alpha \in I_i \cup E_{\text{edge}} \backslash T_{p_{i+1}}} \{\delta_{i+1}(\alpha)\}. \quad (8)$$

Equation (8) indicates that $C_{i+1}$'s MLDP is computed according to the distances between the current piece and the elements of the set $I_i \cup \{S_{\text{left}}, S_{\text{bottom}}, S_{\text{top}}\} \backslash T_{p_{i+1}}$ composed of the pieces already placed, and the three edges of the strip. In this case, both elements of $T_{p_{i+1}}$ used for computing the coordinates of $C_{i+1}$ are excluded, because such a distance is setting equal to zero. Note also that the MLDP is equal to zero when $C_{i+1}$ touches more than two elements.

Figure 1 indicates that the minimum local distance between $C_3$ and the circular pieces already packed, when positioned in $p_3^{(1)}$ and $p_3^{(3)}$, are $\hat{\delta}_{p_3^{(1)}}$ and $\hat{\delta}_{p_3^{(3)}}$ respectively.

Specifically, for a pre-determined ordering of the pieces, the solution procedure starts by positioning the first circular piece $C_1$ at the bottom-left corner, i.e., at the position $(r_1, r_1)$, while the remaining $n - 1$ pieces are successively positioned by using the MLDP rule.

 ISBN: 978-1-61208-109-0 

## IV. THE BSBIS ALGORITHM

Beam search is a tree search strategy where, at each level $\ell$ of the tree, a fixed number of nodes are selected for branching. The aforementioned number (denoted $\omega$) is called the "beam width" (for more details, the reader can refer to Akeb and Hifi [2]).

For the CODP, Akeb and Hifi [2] proposed an algorithm (denoted by BSBIS), which combines two solution procedures: the beam search and the binary interval search. The interval search is denoted by $[\underline{L}, \overline{L}]$, and the role of the beam search procedure is to try to pack all the pieces into the current target rectangle $(L^*, W)$, where $L^* \in [\underline{L}, \overline{L}]$.

Fig. 2. The BSBIS algorithm

**Input.** A node $\eta_\ell$ of level $\ell$ characterized by $I_\ell$, $\overline{I}_\ell$, and $P_{I_\ell}$, the beam width $\omega$, and the bounds of the interval search $\overline{L}$ and $\underline{L}$.
**Output.** A complete feasible packing and the best length $L^*$.

`Initialization phase`
Set $L^* = \overline{L}$, where $L^*$ denotes the best solution found so far;
`Iterative Phase`
`while` $(\overline{L} - \underline{L} > \delta)$ `do`
1) Set $L^* = (\overline{L} + \underline{L})/2$;
2) Set `Feasible` = BS$(\eta_\ell, \omega)$;
3) `if Feasible=true` {*the n pieces have been packed*} `then` set $\overline{L} = L^*$, `otherwise` set $\underline{L} = L^*$;

`enddo`

Different parameters are transmitted to BSBIS (Figure 2): $\eta_\ell$ representing a partial solution (where $\ell$ circles are already placed into the target rectangle), $\omega$ denoting the beam width, and $[\underline{L}, \overline{L}]$ corresponding to the interval search.

At each iteration, (cf. `while` loop), BSBIS calls the BS procedure in order to pack the remaining circles. BS procedure uses a width-first beam search, which is based on the MLDP rule (a detailed version of BS is described in Akeb and Hifi [2]). The interval search is then updated depending on whether a feasible solution is obtained or not (a solution is feasible if the $n$ circles are placed into the target rectangle).

In addition, we note that the interval search needs both lower and upper bound limites. As used in Akeb and Hifi [2], the upper bound $\overline{L}$ is initially generated by applying the *open-strip generation solution procedure* $(\mathrm{OSGSP_a})$ and the lower bound $\underline{L}$ is setting equal to $(\pi \times \sum_{i \in N} r_i^2)/W$.

## V. AN ADAPTIVE ALGORITHM FOR CODP

Herein, we describe an adaptive look-ahead strategy-based algorithm, denoted by A-SEP-MSBS, used for tackling the CODP. The proposed algorithm combines several strategies including the separate-beams search, the multi-start, and the look-ahead ones.

### A. The beam-search look-ahead algorithm (BSLA)

The quality of the results provided by a greedy algorithm depends generally on the criterion used; that can be considered as a decision criterion for such an algorithm. Using a look-ahead strategy may enhance such a decision criterion. Indeed, the purpose of the look-ahead strategy is to try to enrich the criterion of choice, so allowing to increase the quality of the solution provided. Such a strategy can also be viewed as case of the backtracking phase used in branch-and-bound procedures. For instance, such a strategy may explore several paths of the tree search following the quality of the partial solution at hand. It then employs some strategies in order to guide the method to branch towards the best directions. Hence, the strategy used consider a "global evaluation" instead of a "local evaluation". Even the mechanism can require more runtime, but it generally leads to better solutions (as shown in the computational results – Section VI).

The look-ahead strategy, associated with the beam search to solve CODP, leads to the BSLA algorithm detailed in Figure 4. In order to facilitate the readability of BSLA, we present in Figure 3 the look-ahead selection phase (denoted by LASP). We can observe that LASP is called at each step of BSLA (Step 5 of Figure 4).

Fig. 3. The look-ahead selection procedure (LASP)

**Input.** A set $B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$ of $\omega$ nodes and a boolean variable `feasible` initialized to `false`.
**Output.** A feasible solution corresponding to `feasible=true` or a set $B_\omega$ of $\omega$ nodes (those leading to the highest densities through the MLDP packing procedure).

`Initialization phase`
Let $P_{\ell_i}$ be the set of corner positions of node $\eta_\ell^i \in B$;

`Iterative phase`
`for each` node $\eta_\ell^i$ of $B$ `do`
`for each` corner position $p_i \in P_{\ell_i}$ `do`
1) Pack $C_{i+1}$ in $p_i$ and insert the resulting node $\eta_{\ell+1}$ into $B_\omega$;
2) Evaluate the new inserted node $\eta_{\ell+1}$ by placing the remaining circles using the MLDP packing procedure;
3) `if` all circles are placed `then` set `feasible` $=$ `true`, `exit`; `else` assign to $\eta_{\ell+1}$ the density obtained by MLDP;

`enddo`
`enddo`
`Terminal phase`
4) Reduce $B_\omega$ to the $\omega$ nodes that led to the highest densities by using MLDP;
5) Exit with $B_\omega$.

LASP receives two parameters. The first one corresponds to the set $B = \{\eta_\ell^1, ..., \eta_\ell^\omega\}$ containing the nodes of the current level of the search tree. The second parameter `feasible` is an indicator, which takes the value true if the solution reached is feasible, false otherwise. In the `Iterative phase`, the procedure tries to generate the nodes according to the positions of $B$. Such a branching creates the list $B_\omega$ of successors. Second, each node of $B_\omega$ is evaluated according to the MLDP packing (Step 2). Third, if MLDP obtains a feasible solution (Step 3) then the procedure exits with `feasible` = `true`, otherwise the best nodes leading to the highest densities are retuned (Steps 4–5) and so, BSLA (Figure 4) uses these nodes for branching. Of course, the density of a node corresponds to the area of the circles packed divided by the area of the current rectangle $(L^*, W)$.

Figure 4 summarizes the main steps of BSLA. The

 ISBN: 978-1-61208-109-0 

Fig. 4. Beam search look-ahead algorithm (BSLA)

**Input.** A node $\eta_\ell$, the beam width $\omega$, and the bounds of the interval search $\overline{L}$ and $\underline{L}$.
**Output.** A feasible packing and the best corresponding value for the rectangle length ($L_{\text{best}}$).

`Initialization phase`
- Let $B$ and $B_\omega$ be the sets of nodes to be considered and the offspring nodes of the node currently being considered;
- Let $L_{\text{best}}$ be the best length found so far;
- Let `feasible` be a boolean variable;

`Iterative Phase.`
`while` $(\overline{L} - \underline{L} > \delta)$ `do`
1) Set $B = \{\eta_\ell\}$, where $\eta_\ell$ is a starting node of level $\ell$ characterized by $I_\ell$, $\overline{I}_\ell$, and $P_{I_\ell}$;
2) Set $L^* = (\overline{L} + \underline{L})/2$;
3) Set `feasible = false`;
`while` ($B \neq \emptyset$ and `feasible=false`) `do`
4) Set $\ell = \ell + 1$;
5) Set $B_\omega$ = LASP($B$, `feasible`);
6) `if feasible=true then` set $\overline{L} = L^*$ and $L_{\text{best}} = L^*$ `else` set $B = B_\omega$ and $B_\omega = \emptyset$;
`enddo`
7) `if feasible=false then` set $\underline{L} = L^*$;
`enddo`

`Initialization phase`, serves to initialize the interval search, the best length $L_{\text{best}}$ is setting equal to the best length found so far ($L_{\text{best}} = \overline{L}$ otherwise).

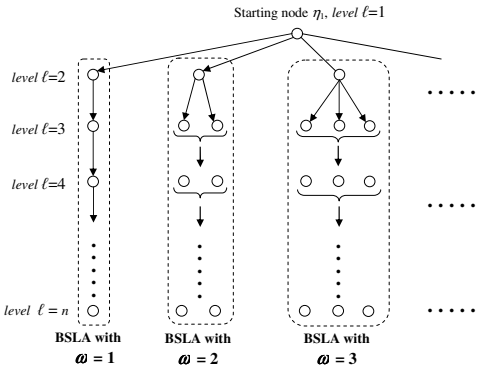
The `Iterative phase` is composed of two loops. The first loop is composed by steps 1, 2, 3 and 7. Steps 1, 2 and 3 serve to initialize $B$ to the current node $\eta_\ell$, to set the current target length of the rectangle $(L^*, W)$ in the dichotomous search and to initialize the indicator `feasible` to false (`feasible` indicates if a feasible packing into the target rectangle $(L^*, W)$ is obtained). The second loop simulates the packing phase. Indeed, Step 4 serves to increment the level $\ell$ before the look-ahead takes place (Step 5, as described in Figure 3) for evaluating each position corresponding to the nodes of the current level. In Step 6, two possibilities may appear: a feasible solution for $(L^*, W)$ is reached (with `feasible =true`), then both lower and upper limits are updated. Otherwise (`feasible=false`) no feasible solution is provided and so, the $\omega$ best expanded nodes representing $B_\omega$ are returned for restarting the set $B$. Finally, in Step 7, `feasible` with the value `false` implies non-existence of feasible packing, the lower bound $\underline{L}$ is then replaced by $L^*$.

BSLA terminates, with the best length $L_{\text{best}}$, when the difference between both limits of the interval search becomes less than or equal to the tolerance gap $\delta$, i.e., when $\overline{L} - \underline{L} \leq \delta$.

### B. The separate-beams strategy

The separate-beams search-based algorithm was proposed by Akeb et al. [13]. The method is mainly based on the separate beams instead of pooled ones (as used in [2]). The mechanism used by the aforementioned strategy can be summarized in Figure 5. First, the root node $(\eta_1)$ corresponds to the first level $(\ell = 1)$ and it contains the starting solution where the first circle is placed at the bottom-left corner of the strip. Second, the search branches out of $\eta_1$ and creates the nodes of level $\ell = 2$. Third, the separate beams are then initiated at the current level. More precisely, the first (best) node initiates a width-first beam search of width $\omega = 1$, the second best node initiates a similar search but width $\omega = 2$, and so on. Hence, the $i^{th}$ node of the current level initiates a beam search of width $\omega = i$. On the one hand, the best nodes, with a high potential to lead to the optimum, do not require an extensive search. The corresponding beam width is then small. On the other hand, the nodes with lesser potentials initiate beam searches with the higher values of the beam width. Note that the width-first beam search initiated by each node at level $\ell = 2$ corresponds to algorithm BSLA (Figure 4) since the look-ahead strategy is used.

Fig. 5. The separate-beams mechanism




### C. Adaptive separate multi-start beam search (A-SEP-MSBS)

A separate-multi-start beam search algorithm, denoted by SEP-MSBS, was first used by Akeb et al. [13]. Indeed, SEP-MSBS combines the separate-beams strategy, described in Section V-B, with a multi-start strategy, which consists in running the algorithm by forcing the first starting circle to pack. The method is then restarted at most $m$ times, where $m$ is the number of different radii of the instance. Herein, we propose to introduce an augmented stage; that is based on the look-ahead strategy. This one is added in order to improve the selection mechanism used at each level of the developed tree. The resulting algorithm, denoted by A-SEP-MSBS, is summarized in Figure 6.

The proposed A-SEP-MSBS algorithm works as follows. First, A-SEP-MSBS starts by ranking the circles in decreasing order of their radii, the best length is initialized to the upper bound $\overline{L}$ (Step 2) and the index $(i_{order})$ of the first circular piece's type to place is initialized (Step 3).

The iterative phase contains two main steps: the generational phase and the improvement phase. The fist phase serves to generate the root node $\eta_1$ by placing the circle $C_{i_{order}}$ into the current rectangle. Second, the algorithm branches out of $\eta_1$ in order to generate the nodes corresponding to the second level $\ell = 2$ (see Figure 5). It then selects the $\omega^{th}$ node from $B$ (Step 7), which corresponds to $\eta_2$ containing two circles already placed in the current rectangle.

The second phase starts by applying BSLA algorithm (Step 8) with $\eta_2$ as an input node; it is used for packing

 ISBN: 978-1-61208-109-0 

Fig. 6. An adaptive separate multi-start beam search (A-SEP-MSBS)

**Input.** A beam width $\omega$.
**Output.** A feasible packing with the best length $L_{\text{best}}$ for the strip.

```
Initialization phase
```
1) Rank the pieces of $N$ in decreasing value of their radii;
2) Set $L_{\text{best}} = \overline{L}$, the best target length found so far;
3) Set $i_{order} = 1$, where $i_{order}$ is the index of the first circular piece's type of the set $M = \{1, ..., m\}$;

```
Iterative phase
Do
```
*Generational step*

4) Generate the node $\eta_1$, characterized by $I_1$, $\overline{I}_1$, and $P_{I_1}$, by placing the first circle $C_{i_{order}}$ inside the current rectangle and let $B = \eta_1$;
5) Branch out of $B$ and generate the list of offspring nodes $B_\omega$;
6) Let $B = \min(\omega, |B_\omega|)$ nodes having the best MLDPs and corresponding to distinct corner positions and reset $B_\omega = \emptyset$;
7) Let $\eta_2$ be the node at position $\omega$ in $B$;

*Improvement step.*

8) Set $feasible$ = BSLA($\eta_2$, $\omega$, $\overline{L}$, $\underline{L}$);
9) `if` $feasible$ = `true then` $\overline{L}$ and $L_{\text{best}}$ are updated if a better length is obtained by BSLA;
10) Set $\underline{L} = (\pi \times \sum_{i=1}^{n} r_i^2)/W$;
11) Set $i_{order} = i_{order} + 1$;

`while` $i_{order} \leq m$;
```
Terminal phase
exit with the best target length L_best.
```

the $n - 2$ remaining circles using the look-ahead strategy (BSLA). Then, two possibilities can be distinguished: (i) `feasible=true` and (ii) `feasible=false`. For the first case, it means that BSLA reaches a feasible solution and so, both $\overline{L}$ and $L_{\text{best}}$ are updated (Step 9). The lower bound $\underline{L}$ of the search is after that reinitialized to the trivial lower bound (Step 10) and another circle is designed (Step 11) for the next restarting of the algorithm

Finally, A-SEP-MSBS terminates with the best length $L_{\text{best}}$, which is reported in column 8 of Table I.

### *D. Adapting the beam width to the look-ahead strategy*

In Akeb and Hifi [2], several tunings have been considered for the principal algorithm BSBIS (Beam Search combined with a Binary Interval Search). Through the aforementioned part, it was noticed that the provided results (the best length of the rectangle) oscillated when the values of the beam width varied. Then, it isn't evident to plan the behavior of such a variation in particular when the values of the beam width increase. In fact, increasing the beam width doesn't guarantee better results, since even the search strategy considers the $\omega$ best ones, the choice is based on the "local evaluation"; that is a *local beam search*. We recall that the look-ahead strategy (as discussed in Section V-A) is applied by considering a "global evaluation"; that is based on the selection of the best nodes leading to the best densities. Such a strategy leads to a *global beam search*, which may contribute to reduce the oscillations of the generated solutions. In fact, we can observe that the length of the target rectangle generally decreases when the beam width increases and so, it is preferable to run the look-ahead-based algorithm with some largest values of $\omega$. Our limited computational results showed that running A-SEP-MSBS algorithm with a beam width starting from 10 and incremented by 5 (i.e., $\omega = 10+5*k,\ k \in \mathbb{N}$), provided a good compromise between the quality of the provided solutions and the computation time.

## VI. COMPUTATIONAL RESULTS

The objective of the computational investigation is to assess the performance of the proposed algorithm A-SEP-MSBS by comparing its solution quality to the best known results in the literarture. A-SEP-MSBS was coded in C language and run on a 3-GHz processor and 256 MB of RAM.

Two sets of instances were used in the comparison. The first set contains the six instances taken from Stoyan and Yaskov [10] namely SY1–SY6. Each instance contains between 20 and 100 circles. The second set of instances is composed of twelve problems taken from Akeb and Hifi [2], and are obtained by concatenating the six original problems of the first set. The problems of the second set are identified as SY12, SY13, SY14, SY23, SY24, SY34, SY56, SY123, SY124, SY134, SY234, and SY1–4 and contain between 45 and 200 circles. Thus, varying the number of circles $n$ from 20 (small size) to 200 (large size) reflects objectively the behavior of the proposed algorithm.

TABLE I
SOLUTION QUALITY OBTAINED BY ALGORITHM A-SEP-MSBS

| | | | | Best | SEP-MSBS | | A-SEP-MSBS | |
|---|---|---|---|---|---|---|---|---|
| Inst. | $n$ | $m$ | $W$ | $L$ | $L$ | $\omega^*$ | $L$ | $\omega^*$ |
| SY1 | 30 | 30 | 9.5 | 17.2070 | 17.2070 | 50 | **17.0954** | 20 |
| SY2 | 20 | 20 | 8.5 | 14.4867 | 14.5287 | 24 | **14.4548** | 15 |
| SY3 | 25 | 25 | 9 | 14.4176 | 14.4616 | 44 | **14.4017** | 70 |
| SY4 | 35 | 35 | 11 | 23.4921 | 23.4921 | 66 | **23.3538** | 10 |
| SY5 | 100 | 99 | 15 | 36.0796 | 36.1818 | 22 | **36.0061** | 10 |
| SY6 | 100 | 98 | 19 | 36.7197 | 36.7197 | 26 | **36.6629** | 10 |
| SY12 | 50 | 48 | 9.5 | **29.6837** | **29.6837** | 61 | 29.8148 | 20 |
| SY13 | 55 | 54 | 9.5 | **30.3705** | **30.3705** | 68 | 30.4547 | 15 |
| SY14 | 65 | 65 | 11 | 37.8518 | 37.8518 | 63 | **37.7244** | 10 |
| SY23 | 45 | 45 | 9 | **27.6351** | **27.6351** | 89 | 27.7574 | 30 |
| SY24 | 55 | 54 | 11 | **34.1455** | **34.1455** | 49 | 34.1511 | 15 |
| SY34 | 60 | 59 | 11 | **34.6376** | 34.6859 | 43 | 34.6744 | 10 |
| SY56 | 200 | 193 | 19 | **64.7246** | 65.2024 | 6 | 64.7876 | 10 |
| SY123 | 75 | 72 | 9.5 | **42.9931** | 43.0306 | 25 | 43.0930 | 15 |
| SY124 | 85 | 82 | 11 | 48.8411 | 48.8411 | 35 | **48.6101** | 15 |
| SY134 | 90 | 88 | 11 | 49.3254 | 49.3362 | 27 | **49.2739** | 15 |
| SY234 | 80 | 78 | 11 | 45.5576 | 45.6115 | 39 | **45.4586** | 15 |
| SY1–4 | 110 | 105 | 11 | **60.0564** | **60.0564** | 25 | 60.3346 | 10 |

The results obtained by A-SEP-MSBS are compared to those obtained by SEP-MSBS [13] and to the best known results in the literature obtained either by BSBIS [2] or by the maximum hole degree heuristic (algorithms B1.0 and B1.5) [4].

For an accurate comparison, the maximum beam width value $\omega$ does not exceed 100 (the same limit was used in [2], [13]) and the cumulative computation time is fixed to thirty hours for all the algorithms (B1.0 and B1.5 were also ran with this time limit). So each algorithm may stop after attaining the beam width limit or when it exceeds the fixed computation

 ISBN: 978-1-61208-109-0 

time. Not that $\omega = 10 + 5 * k, k \in \mathbb{N}$ for algorithm A-SEP-MSBS as explained in Section V-D, and $\omega \in [1, .., 100]$ for BSBIS and SEP-MSBS.

Table I displays the results provided by the proposed algorithm A-SEP-MSBS. The solution quality is compared to the solution obtained by several other algorithms. Columns 1 and 2 of Table I indicate the instance name and its size. Column 3 shows the number of circle types in the instance, i.e., the number of different radii (this parameter is used by the multi-start strategy). Column 4 indicates the width of the strip $(W)$ for each instance. Column 5 reports the best known solution in the literature obtained either by BSBIS [2], SEP-MSBS [13], or by the MHD heuristic (algorithms B1.0 and B1.5) [4]. Column 6 indicates the best length of the strip corresponding to the SEP-MSBS algorithm, and Column 7 gives the beam width $\omega^*$ used to obtain this solution. Finally, Columns 8 and 9 display the same information than Columns 6 and 7 but for the proposed algorithm A-SEP-MSBS.

The results of Table I indicate that algorithm A-SEP-MSBS improves the best known results in ten occasions out of eighteen. It improves SEP-MSBS in twelve occasions. Note that the new algorithm (A-SEP-MSBS) improves all the best known solutions for the first set of instances (SY1–SY6) with an average percentage of improvement equal to 0.32%. This percentage is computed as $\frac{L_{\text{best}} - L}{L_{\text{best}}} \times 100\%$, where $L_{\text{best}}$ is the best known solution in the literature (Column 5) and $L$ is the length obtained by A-SEP-MSBS (Column 8). When including the second set of instances, this improvement becomes 0.05%. It is also to note that A-SEP-MSBS improves SEP-MSBS in twelve occasions with 0.16% in average.

The computation time is not reported in Table I. This is because the time limit of thirty hours was attained by A-SEP-MSBS for all the instances. This phenomenon is due to the use of the look-ahead strategy, which is very time-consuming. The time limit (thirty hours) was also attained by the SEP-MSBS algorithm [13] for thirteen instances on eighteen (except for SY1, SY2, SY3, SY4 and SY23), i.e., when $n > 45$.

Table I shows that A-SEP-MSBS (even if it performs better than the other methods) is especially efficient for small instances for which the maximum attained values of $\omega$ (according to the time limit) is greater than for the larger instances. So for these instances, A-SEP-MSBS can be used. For larger instances, the standard beam search algorithm [2] often obtains better results. In order to apply A-SEP-MSBS on the larger instances, the best solution would be the parallelization of this algorithm in order to explore more search space with the same run time limit.

Fig. 7. Solution of A-SEP-MSBS on SY1 ($n = 30$ and $L = 17.0954$)

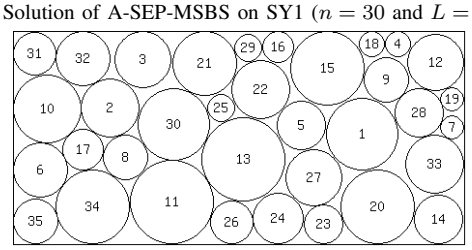



Figure 7 displays the solution obtained by algorithm A-SEP-MSBS on the first instance (SY1, $n = 30$). The previous best known solution was $L = 17.2070$ obtained by SEP-MSBS [13], and the new solution is $L = 17.0954$, i.e., an improvement of 0.65%.

## VII. CONCLUSION AND FUTURE WORK

In this paper an adaptive look-ahead strategy-based algorithm is proposed for approximately solving the circular open dimension problem. The algorithm combines several strategies: beam search, look-ahead, multi-start, as well as a separate beams strategy. The computational investigation shows that the proposed algorithm, for a set of benchmarks instances taken from the literature, is able to improve several best known solutions. The results showed also that the look-ahead strategy needs more runtime for large size instances. Then, we think that the parallel implementation of such an approach may enlarge the search space and so, improve the quality of the solutions. A parallel algorithm may also allow processing larger industrial problems (instances with more than one thousand of circles for example) in a reasonable computation time.

## REFERENCES

[1] G. Wäscher, H. Haussner, and H. Schumann, "An improved typology of cutting and packing problems", *European Journal of Operational Research*, vol. 183, 2007, pp. 1109–1130.

[2] H. Akeb and M. Hifi, "Algorithms for the circular two-dimensional open dimension problem", *International Transactions in Operational Research*, vol. 15, 2008, pp. 685–704.

[3] M. Hifi and R. M'Hallah, "A hybrid algorithm for the two-dimensional layout problem: the cases of regular and irregular shapes", *International Transactions in Operational Research*, vol. 10, 2003, pp. 195–216.

[4] W. Q. Huang, Y. Li, H. Akeb, and C. M. Li, "Greedy algorithms for packing unequal circles into a rectangular container" *Journal of the Operational Research Society*, vol. 56, 2005, pp. 539–548.

[5] R. L. Graham, B. D. Luboachevsky, K. J. Nurmela, and P. R. J. Östergård, "Dense packings of congruent circles in a circle", *Discrete Mathematics*, vol. 181, 1998, pp. 139–154.

[6] W. Q. Huang, Y. Li, C. M. Li, and R. C. Xu, "New heuristics for packing unequal circles into a circular container", *Computer and Operations Research*, vol. 33, 2006, pp. 2125–2142.

[7] M. Hifi and R. M'Hallah, "A dynamic adaptive local search based algorithm for the circular packing problem", *European Journal of Operational Research*, vol. 183, 2007, pp. 1280–1294.

[8] H. Akeb, M. Hifi, and R. M'Hallah, "A beam search based algorithm for the circular packing problem", *Computers & Operations Research*, vol. 36, 2009, pp. 1513–1528.

[9] J. A. George, J. M. George, and B. W. Lamar, "Packing different-sized circles into a rectangular container", *European Journal of Operational Research*, vol. 84, 1995, pp. 693–712.

[10] Y. G. Stoyan and G. N. Yaskov, "Mathematical model and solution method of optimization problem of placement of rectangles and circles taking into account special constraints", *International Transactions in Operational Research*, vol. 5, 1998, pp. 45–57.

[11] M. Hifi and R. M'Hallah, "Approximate algorithms for constrained circular cutting problems" *Computers and Operations Research*, vol. 31, 2004, pp. 675–694.

[12] E. G. Birgin, J. M. Martinez, and D. P. Ronconi, "Optimizing the packing of cylinders into a rectangular container: A nonlinear approach", *European Journal of Operational Research*, vol. 160, 2005, pp. 19–33.

[13] H. Akeb, M. Hifi, and S. Negre, "An augmented beam search-based algorithm for the circular open dimension problem", *Proceedings of the CIE 2009, International Conference on Computers & Industrial Engineering*, Troyes, France, July 2009, pp. 372–377.

# Platform Adaptation of Mashup UI Components

Andreas Rümpel, Ken Baumgärtel, and Klaus Meißner
*Chair of Multimedia Technology*
*Technische Universität Dresden*
*01062 Dresden, Germany*
{*andreas.ruempel,ken.baumgaertel,klaus.meissner*}*@tu-dresden.de*

***Abstract*—Modern *user interface mashups* combine web-based backend services and a composite user interface. While the former can be used in different runtime platforms without further ado, user interface components are rendered platform-dependently. Creating and maintaining multiple implementation variants for different technologies and communication interfaces implies high development costs. This paper introduces a generic platform adaptation concept of such visual mashup components. The practicability of the proposed concept is demonstrated by implementing the adapter for a subsistent integration infrastructure and component model.**

***Keywords*-user interface integration; platform adaptation; mashup components; composite web applications**

## I. Introduction and Motivation

Old-fashioned ways to create web applications presupposed a specific execution platform or framework. This includes server-centric ones, like portal servers, those running completely on the web client, e. g., following the *Thin Server Architecture* [1], and hybrid ones like *Eclipse Rich Ajax Platform (RAP)* [2]. When building composite web applications for those platforms, the availability of compatible UI components is essential. Such UI components cannot be reused in other runtime environments easily, because different web platforms and runtime frameworks have specific technology-induced requirements and behavior concerning their structure, language and deployment mode and thus impede a cost-efficient and fast development process. This platform heterogeneity implies a tedious manual adjustment and migration process of existing components to be executed in alternative environments. We understand these runtime platforms as one special kind of context called *integration context*. Supporting various integration contexts can be regarded as a new quality of adaptivity of such applications, driven by their components.

In most composite systems, *component interface descriptions* represent the formal realization of a *component model*. They are used to facilitate the application's internal communication. The existence of such component interface descriptions is an essential precondition of our adaptation concept. In contrast to conventional web-based services, user interface building parts apparently cannot be platform-independent per se, because they are always rendered based on concrete code, i. e., they mostly contain or generate HTML code in conjunction with JavaScript. We define *platforms* including web runtime architectures and UI toolkits as well, which are already present in a great diversity. This indicates that a platform adaptation mechanism is inevitable against the background of using UI components in a multi-platform context. Fortunately, the major part of one UI component can be generated automatically, e. g., through derivation from an interface description during platform adaptation.

This paper provides a novel concept introducing a platform adapter for UI components of composite web applications. It takes existing user interface components conforming to a specific component model and developed for a specific runtime environment as adaptation input. Based on templates, it performs platform adaptation by wrapping and generating the parts relevant to the target runtime platform.

The remaining paper is structured as follows. Section II analyzes and defines prerequisites regarding the component model in distributed web UI scenarios. Section III presents our concept of platform adaptation for UI components. Afterwards, in Section IV, the proposed concept is applied to the CRUISe UI composition infrastructure [3], which was developed in parallel. Implementation details are given in Section V. In Section VI, related work is outlined. Finally, Section VII concludes this paper.

## II. Component Model and Prerequisites

We consider UI components in the scope of web-based composite applications, causing the need for a client-side part to be rendered by a web browser engine. This client-side part, generically created or not, consists either of HTML and JavaScript, or employs a browser plug-in technology like Adobe Flash or Microsoft Silverlight. To specify an adaptation concept, a discussion of valid combinations of source UI component implementation architectures and target platforms is necessary. The following comparison elucidates different commonly used UI component implementation techniques and derived consequences for the adaptation process with special focus on client-server distribution.

***Client-side JavaScript**:* Components using only plain HTML and JavaScript frameworks are well-suited to be adapted to other JavaScript frameworks, hybrid or server-side technologies because of their easy to handle declarative

 ISBN: 978-1-61208-109-0 

```
<interface xmlns:dt="http://uis.dyndns.org/datatypes">
   <event name="singleSelection">
      <parameter type="dt:Address" name="Address"/>
   </event>
   <event name="multiSelection">
      <parameter type="dt:AddressList" name="AddressList"/>
   </event>
   <event name="exportSelected">
      <parameter type="dt:AddressList" name="AddressList"/>
   </event>
   <operation name="addRecord">
      <parameter type="dt:Address" name="Address"/>
   </operation>
   <operation name="updateRecord">
      <parameter type="dt:Address" name="Address"/>
   </operation>
</interface>
```

Listing 1. Interface description of an address list UI component

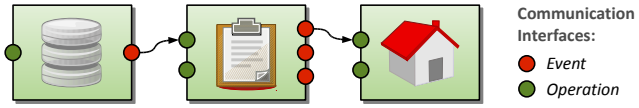



Figure 1. User interface components in a mashup application

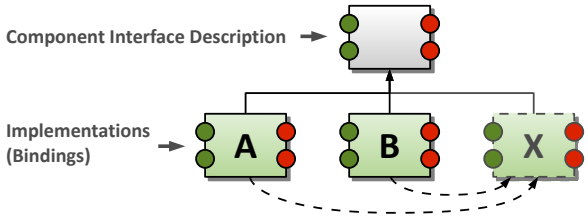



Figure 2. Adaptation as derivation of new UI component binding

and scripted nature. In the latter two cases, additional component parts on the server side have to be generated.

***Browser plug-in:*** Adobe Flash, Microsoft Silverlight and other browser plug-in technologies are also client-side approaches. They are scriptable using a JavaScript bridge and then can be adapted to the same as above.

***Hybrid techniques:*** They are suitable to be adapted under certain conditions, if the UI-relevant part can be extracted like in owner-drawn RAP widgets [4]. In this case, the output format range is like the above.

***Server-side (binary and scripted):*** Server-centric component implementations require a code analysis of source components, if technologies like portlets are used and an extraction technique for server-side scripting approaches, like Java Server Pages. Possibilities supporting server-side input components are very limited, because binary components or component parts would have to be provided as source code.

Obviously, the ability of handling implementations of available source components is a crucial factor when performing component adaptation. An important cornerstone of our adaptation concept is the use of a *component model*, formally describing the communication interfaces of components. To exemplarily illustrate the contents of such a model, a snippet of a mashup component description used in CRUISe (cf. Section IV) is shown in Listing 1. It describes the interface of an *AddressList* UI component. The interface model used here comprises *events* and *operations* to establish an event-based communication within the application. Three events (*singleSelection*, *multiSelection* and *exportSelected*) and two operations (*addRecord* and *updateRecord*) with corresponding data types are provided by this component. Using those interface parts, the addresses can be read out or manipulated.

In Figure 1, the explained component is shown in a simple real estate management application context. New addresses can be added to **AddressList** (middle, same interface as outlined before) by another component, e. g., a **Building-Database** (left), providing new address entries using the operation *addRecord*. Once an address entry is selected in the list (*singleSelection*), the building corresponding to the selected address is visualized by an **ImageViewer** component (right). The whole communication (arrows) is realized by an event-based interface in a publish-subscribe fashion.

We conceive component implementations as *bindings* for an abstract component interface description. Accordingly, different bindings of the same interface are exchangeable against each other within an application. Thus, the proposed platform adapter can be regarded as a generator for new bindings, matching the previously unsupported integration context to adapt to. As illustrated in Figure 2, an existing binding (e. g., *A*) for another integration context is taken as a source of adaptation to derive the new one (*X*) utilizing platform-specific creation rules (arrows). These creation rules have to be specified once per platform to be supported.

## III. UI Component Platform Adapter

Based on the previously stated prerequisites and conditions, we now present our concept of platform adaptation of UI components proposing a multi-staged platform adapter to be used within web integration architectures. The adaptation workflow is executed for one UI component at a time. Starting point and input for the adapter are the component's abstract *interface description* and one of its *bindings*. As shown in Figure 3, an abstraction stage precedes the template-based generation of the target implementation to transform non-matching input bindings into a platform-compatible one. The service, the platform adapter offers to the UI integration system, can be summarized as the provision of a previously unavailable binding for a specified UI component. Beside a list of source bindings and the component's interface description, the adapter requires an identifier of the desired target platform as input. While the support of different platforms mainly affects the second adaptation stage, the used abstract component interface model has to be specified for the whole adapter. Hence, different component models employing alternative communication paradigms are fine, as long as they conform to the outlined prerequisites (cf. Section II). The two stages are now described in detail.

 ISBN: 978-1-61208-109-0 

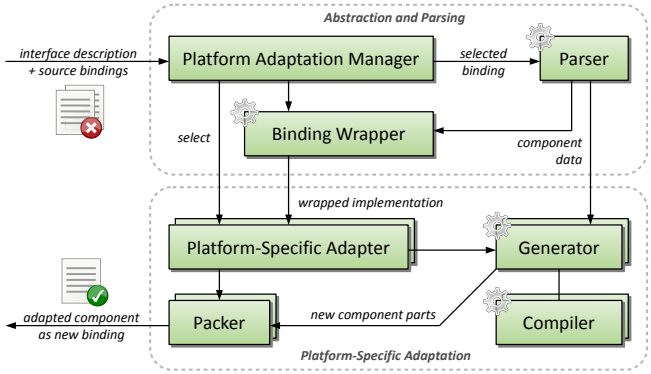



Figure 3. Platform adapter architecture

```
<binding platform="javascript-client-runtime-platform">
   <interface>http://uis.dyndns.org/uic?class=addresslist</interface>
   <dependencies>
      <dependency uri="http://uis.dyndns.org/addresslist.js" language="js"/>
   </dependencies>
   <constructor>@:instance@ = new ui.component.AddressList()</constructor>
   <destructor>@:instance@.remove()</destructor>

   <eventsink event="event.singleSelection">
      <register>@:instance@.on('singleSelection',
                @event:event.singleSelection@)</register>
   </eventsink>
   <eventsink event="event.multiSelection">
      <register>@:instance@.on('multiSelection',
                @event:event.multiSelection@)</register>
   </eventsink>
   <eventsink event="event.exportSelected">
      <register>@:instance@.on('exportSelected',
                @event:event.exportSelected@)</register>
   </eventsink>
   <invocation operation="op.addRecord">
                @:instance@.addRecord(@parameter:Address@)</invocation>
   <invocation operation="op.updateRecord">
                @:instance@.updateRecord(@parameter:Address@)</invocation>
</binding>
```

Listing 2. Binding for a JavaScript *AddressList* component

### A. Abstraction and Parsing

In this stage, the analysis of the components' interface description and the derivation of an internal data model of the input UI component binding are performed. The *Platform Adaptation Manager (PAM)* controls the whole adaptation process. In doing so, it selects one of a list of available source bindings to proceed further steps. As shown in Listing 2, bindings are represented by an XML description for a specific `platform`, referencing needed `dependencies` and containing the mapping to language-specific constructs, as the JavaScript snippets illustrate. The depicted binding conforms to the *AddressList* component (cf. Listing 1), telling how the implementation should be addressed through `invocation` of *operations* and `eventsinks` for publishing *events*.

The binding is analyzed by a *parser* to create an internal interface model, which is forwarded to the *binding wrapper*. Its task includes the resolution and download of the component's dependencies. This step is necessary, because components differ in their style of referencing and including resources, such as images, styling information or script files. Based on the collected component data, a common representation for further processing in the second adaptation stage is gained.

### B. Platform-Specific Adaptation

The wrapped component implementation represents the input for a *Platform-Specific Adapter (PSA)*, which is selected by the *PAM* depending on the target platform identifier specified. A concrete PSA controls the *template-based* creation of all parts of the new UI component implementation. Different *generators* and *compilers* are invoked by the PSA to perform this task. The complexity of implementation artifacts to be generated depends on the target platform (cf. Section II). Having all parts generated, a *packer* brings them into a deliverable format, creates an XML file describing the binding declaratively (cf. Listing 2) and links resources within the binding document.

For supporting a large range of platforms, a flexible internal component handling is required. The adaptation to a client-side platform considers inter-component communication and integration on the client, while a server-side platform performs this on the server. Hence, the adapted UI component is divided into different parts along the lines of a design pattern called *Half-Object plus Protocol (HOPP)* [5]. This division enables the partitioning of provided functionality of the UI component depending on the target platform and introduces the principle of *Model View Controller (MVC)* meeting the demands of adaptable UI components. With this conceptual division, each part of the component can be adapted effectively, so that it will fit best into the target platform. If, e. g., a component for a client-server distributed integration context with server-side communication is targeted, HOPP helps to build a compatible server-side interface and to connect it with the corresponding client-side part of the component. Figure 4 shows the distributed nature of the *AddressList* component outlined in Listings 1 and 2 for a client-server distributed integration context. The server-side part represents the model, the client-side part the view and another part, called *Synchronization Controller*, adopts the implementation of the used protocol and represents the controller, or bridge, between both other parts. This distribution causes the server-side part holding a reference to the wrapped, previously client-side UI component and is therefore a *proxy*. The input (green) and output (red) access points enable component communication. This communication can also be parallel, on client and server, yielding UI components enabled to communicate offline, synchronizing their corresponding parts and persisting their states when a network connection is established.

The splitting, i. e., distribution of the component, is not mandatory. It depends on the given component technology and the architecture of the target platform. Thus, it is possible to have only a client-side wrapper (e. g., for a thin server runtime) without the other parts. Assuming a distributed

 ISBN: 978-1-61208-109-0 

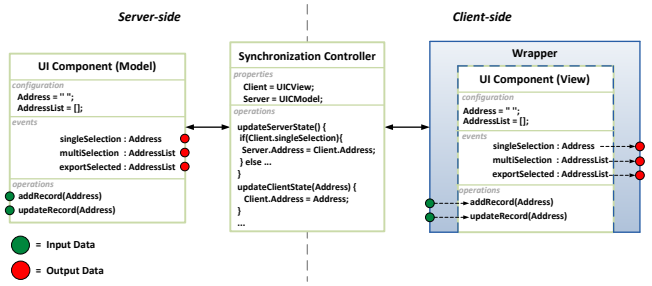



Figure 4. Distributed adapted UI component for a thin client platform

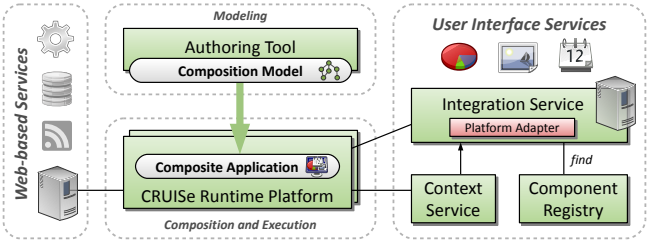



Figure 5. CRUISe infrastructure

integration context, like Eclipse RAP [2], a distributed component is generated. It consists of a package containing the server-side model, the client-side JavaScript component and the so called *Life Cycle Adapter* to synchronize both. In other situations, e. g., a JavaScript-based integration context, it is sufficient to create a JavaScript wrapper. Thus, a suitable workflow is achieved, that covers many platform-specific possibilities.

### C. Adapter Usage and Integration

At least one proper PSA per target platform has to be provided to make use of this adapter. Therefore, it is the PAM's responsibility to choose the right PSA at runtime. To develop Platform-Specific Adapters, adaptation experts require a high knowledge of the integration contexts to cover. If there is no adequate PSA available, the adaptation is aborted and the PAM cannot deliver a new binding. Runtime platform providers are in charge of creating and maintaining their PSAs. The adaptation process can be called at application runtime, at UI integration time (cf. Section IV), or at component deployment time in a component registry, which denotes a higher performance. Moreover, the adapter itself could be outsourced and act as an external service, the integration system could use. A similar way of modularization could be applied to the PSA to yield *Platform Adapation as a Service*. This would achieve the total separation from an explicit integration infrastructure.

## IV. Service-Based UI Integration System

The *CRUISe system* [3] facilitates the integration of service-based user interface building parts, called *User Interface Services (UIS)*, to create *user interface mashups*. This section describes how CRUISe is utilized to provide a UI integration infrastructure hosting the platform adapter.

### A. Architectural Overview

As shown in Figure 5, using a platform-independent *composition model* and a model-to-code transformation, one application can be executed on different kinds of distributed runtime platforms. The composition model is transformed into a platform-specific composite CRUISe application (green arrow). Within this model, UI component interfaces and their communication relations are specified. The generated platform-specific application contains binding points for UI components, enabling binding and exchange at runtime. Thus, it is called *application skeleton*. The interface descriptions of the components to be integrated are organized in classes to modularize them and facilitate the implementation of multiple bindings. Referencing such interface classes (cf. interface description in Listing 1), all information needed to integrate UI components in a service-oriented fashion is provided. Taking those descriptions and a platform identifier, an integration request is created and sent from the runtime platform to the *Integration Service*, which performs further steps of retrieval, selection and adaptation, utilizing the *Component Registry* and external *Context Services*. The ready-to-integrate UI component is then returned to the runtime environment. A crucial condition for this workflow is the availability of platform-compatible component implementations, i. e., bindings, to be delivered. Hence, only existing bindings can be provided at this point without a platform adapter. Since the binding selection and ranking process is executed with the *Integration Service* in charge, our adapter is realized there as a module.

### B. Integration Workflow

An integration request is sent from the runtime environment to the *Integration Service* carrying component interface descriptions and a platform identifier. Interface descriptions are extracted from the application skeleton, the platform is determined by the runtime platform itself based on a defined set of identifiers. By executing an *Integration Task*, this request is forwarded to the *Component Registry*, i. e., it is asked for matching UI component implementations of the UI component, that is subject to integration. The further workflow can be divided into two cases.

***Platform binding available:*** The registry discovers bindings matching the component interface and the platform. Then a list of compatible bindings is returned to the *Integration Service* and no platform adaptation is needed. Next, a ranking module performs a context-based ranking process, yielding the most appropriate UI component in respect of the given requirements.

***Platform binding unavailable:*** The registry can find bindings matching the interface description, but not the

 ISBN: 978-1-61208-109-0 

platform parameter. As a consequence, the registry sends back an empty list. Next, the integration service sends a new request, but without restricting the platform. A list of bindings is returned, which are not matching the initially desired platform. Afterwards, the *Integration Service*, knowing that these bindings are not platform-compatible, can continue to forward this list to the platform adapter. There, the *Platform Adaptation Manager* receives the list and performs further operations described in Section III. The employment of a ranking module is obviously not needed in that case.

Thus, platform adaptation is added as a further step to the CRUISe integration task. Its invocation is depending on the output bindings of the *Component Registry*. In both cases, the finally selected binding is returned to the runtime environment, which performs its integration into the composite application. The new binding may comprise different new files, that were created during adaptation. To allow the runtime environment to access them, they have to be hosted under a specific address, which is referenced within the binding XML document. This kind of hosting is provided by the *Integration Service*.

While processing this integration workflow, some exceptional situations could occur. It is possible that the registry cannot find any bindings, even when neglecting the platform parameter, but then a UI component without implementations would exist. This situation should be avoided by the component developers. Incorporating the proposed concept, the model-based development and UI integration approach in CRUISe facilitates the execution of one mashup application in different runtime platforms.

## V. Implementation

Since a concrete component model is required to implement the platform adapter, the CRUISe-specific model was chosen to fit into the integration infrastructure described in Section IV. For the template-based component part generation we used *Apache Velocity* [6] as a template engine. Templates are provided with the selected PSA for each supported target platform. Each template contains *place holders*, using *Velocity Template Language*. They are filled by a *template engine* with information from the extracted component data (Figure 6). A place holder defines a variable part in the template. It is expressed by the $-symbol for accessing data objects and optionally embedded in statements given by the #-symbol, e. g., to handle loops. With both constructs, each template can be described effectively. In Figure 6, a new instance of the *AddressList* component and the definition of the component's operations are represented by place holders (cf. Listing 2).

The step of generating the component's parts includes the download of and reference to source component artifacts, gained from the binding description, and the optional compilation of created source code after template processing. If, e. g., a component part based on Java should be created, the

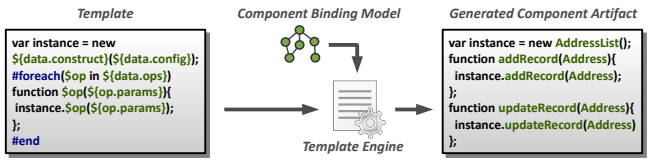



Figure 6. Template-based component artifact generation

PSA would call a Java compiler to generate bytecode out of the Java source code. For such typed programming languages, the right communication data types in the templates have to be defined. The used component model defines data types with XML Schema. Finally, an additional generation step of the target component structure and packaging to the target format is executed. In the case of RAP, it is packed as a *Java Archive* to provide an *OSGi bundle*. An RAP-based runtime can load those bundled components and integrate them into an RAP application. Hence, the package contains artifacts of the original UI component, its resource dependencies, like optional CSS, Flash or JavaScript files, wrapping code and the new XML binding description, generated by a special template. Finally, the new binding is returned and can be committed to the hosting integration infrastructure.

## VI. Related Work

While the attention for mashup development has been overwhelming, only few preliminary work has been done in the field of mashup component adaptation. Often, a sufficiently large pool of perfectly matching UI components is assumed, but compatibility issues initiated by different component models and runtime platforms are neglected.

In [7], a wrapping mechanism of generic web applications is described by using a predefined component model to provide them for further composition. The transformation of web applications is done using *event annotations* to yield mashup components conforming to the component model. A generic wrapper structure enables the support of componentization at runtime. An API is created, which generates events, enacts operations, allows the instantiation and renders the component's UI. In contrast to our approach, they propose the engineering of new components by using existing web pages. No source component model is presumed. The goal is to create new mashup components for further composition. This process is performed by a component developer with a special tool, while our adapter creates new component implementations automatically, showing the huge relevance of a predefined component model.

A similar approach of adapting existing UI building parts is also used in the *CAMELEON* project [8]. It identifies three steps to re-engineer a UI to other contexts. Multiple levels of abstraction provide the step-wise process of generation [9]. First of all, UI interaction objects of a web page are detected, while a user provides feedback on the objects he

 ISBN: 978-1-61208-109-0 

is interested in. Next, a presentation model is created out of them. After that, the model is transformed into another presentation model of the new context. It can be manually modified before creating the final user interface. In contrast to CAMELEON, we strive for an automatic process without any user feedback at runtime. Furthermore, it is hardly applicable to analyze a UI without the knowledge of its component model. Thus, the kind of component introspection and re-engineering like in CAMELEON is not suitable in our case. Further, to support a service-oriented UI provision, parsing is only applicable at the level of component interface description.

*Mixup* elaborates the usage of component adapters for the integration of presentation components [10]. They aim for the support of heterogeneous components with different underlying technologies. A lightweight middleware is used to instantiate these adapters to allow the communication of UI components at runtime. The adapter locates the right implementation and instantiates the component. It identifies and allows access to events, operations and properties and performs data type mapping. To apply this adapter concept, a meta-language facility like reflection is needed. If this is not satisfied, a generic adapter cannot be built. In that case, the implementation of individual wrappers for each component is needed. We also use a component wrapper, which is part of the adapted UI component itself. For creating this wrapper we use an abstract interface description of the component as an input. This is very important for encouraging the independence of possible runtime systems as the adapter does not need to be part of them.

## VII. Conclusion and Future Work

In this paper, we presented a platform adaptation concept for UI components, which are used within web mashup applications. Therefore, we took the existence of a *component model* as a starting point and treated component implementations as *bindings* to their interface descriptions realizing this model. We regarded the platform adaptation process as a template-based generation of further bindings to a given component interface supporting different runtime platforms. By transferring the platform adapter to an existing user interface integration infrastructure, we showed the operational capability of our concept, implementing *Platform-Specific Adapters*, e. g., for a runtime environment based on Eclipse Rich Ajax Platform. All in all, we are able to employ web UI components in a multi-platform scope.

The current approach does not automatically involve similarities between different integration contexts. To this end, a classification of integration contexts could help to derive PSA implementations covering new platforms. Thus, a semi-automatic adaptation workflow for unknown platforms similar to known ones would be possible, being a very useful tooling support for platform refactoring and version management. We further plan to investigate how generated bindings, i. e., adapted components, can be saved and then made available to a component repository to be used in further integration tasks. An additional research subject will cover the extension of the UI component adaptation concept to a general mashup component adaptation concept. Prerequisites are a uniform mashup component model and analyses of the adaptation and exchangeability needs of components such as data type converters or service proxies.

## Acknowledgements

The authors wish to thank CAS Software AG for providing support by developing a prototypical RAP-based runtime platform to be used as an integration context.

## References

[1] S. Pietschmann, J. Waltsgott, and K. Meißner, “A thin-server runtime platform for composite web applications,” in *Proceedings of the 5th International Conference on Internet and Web Applications and Services (ICIW 2010)*. IEEE CPS, May 2010.

[2] B. Muskalla and R. Sternberg, “RCP goes web 2.0,” *Eclipse Magazin*, vol. 12, Oct. 2007.

[3] S. Pietschmann, M. Voigt, A. Rümpel, and K. Meißner, “CRUISe: Composition of rich user interface services,” in *Web Engineering*, ser. Lecture Notes in Computer Science, vol. 5648/2009. Springer, Jun. 2009, pp. 473–476.

[4] F. Lange, *Eclipse Rich Ajax Platform: Bringing Rich Clients to the Web*. Apress, Dec. 2008.

[5] G. Meszaros, “Pattern: Half-object plus protocol (HOPP),” in *Pattern languages of program design*, J. O. Coplien and D. Schmidt, Eds. Addison-Wesley Longman, May 1995, pp. 129–132.

[6] Apache Software Foundation, “The apache velocity project,” accessed: 2010-08-26. [Online]. Available: http://velocity.apache.org

[7] F. Daniel and M. Matera, “Turning web applications into mashup components: Issues, models, and solutions,” in *Web Engineering*, ser. Lecture Notes in Computer Science, vol. 5648/2009. Springer, Jun. 2009, pp. 45–60.

[8] L. Bouillon, J. Vanderdonckt, and K. C. Chow, “Flexible re-engineering of web sites,” in *Proceedings of the 9th international conference on Intelligent user interfaces (IUI ’04)*. ACM, Jan. 2004, pp. 132–139.

[9] G. Calvary, J. Coutaz, D. Thevenin, Q. Limbourg, L. Bouillon, and J. Vanderdonckt, “A unifying reference framework for multi-target user interfaces,” *Interacting with Computers*, vol. 15, no. 3, pp. 289–308, Jun. 2003.

[10] J. Yu, B. Benatallah, R. Saint-Paul, F. Casati, F. Daniel, and M. Matera, “A framework for rapid integration of presentation components,” in *Proceedings of the 16th international conference on World Wide Web (WWW ’07)*. ACM, May 2007, pp. 923–932.

 ISBN: 978-1-61208-109-0 

# Adapting Abstract Component Applications Using Adaptation Patterns

Imen Ben Lahmar*, Djamel Belaïd*, and Hamid Muktar†
*Institut Telecom; Telecom SudParis, CNRS UMR SAMOVAR, Evry, France
Email: {imen.ben_lahmar, djamel.belaid}@it-sudparis.eu
†National University of Sciences and Technology, Islamabad, Pakistan
Email: hamid.mukhtar@seecs.edu.pk

***Abstract*—Using a component-based approach, applications can be defined as an assembly of abstract components, requiring services from and providing services to each other. At the time of execution, they are mapped to the concrete level after identifying the deployed components. However, several problems can be detected at init time that prevent the mapping to be achieved successfully, e.g., heterogeneity of connection interfaces. Moreover, applications in pervasive environment are challenged by the dynamism of their execution environment due to, e.g., users and devices mobility, which make them subject to unforeseen failures.**

**Both of these problems imply mismatches between abstract and concrete levels detected at init time or during the execution. Therefore, abstract applications have to be adapted to carry out their mapping and their execution.**

**In this article, we propose a new dynamic structural adaptation approach for abstract applications. Our approach is based on adaptation patterns that provide solutions to the captured mismatches between abstract and concrete levels. We also compare and contrast our approach with the existing ones concluding that our approach is not only generic, but it is also applicable both at init time and at runtime.**

***Index Terms*—Adaptation patterns, adapter template, abstract application, mismatch, pervasive environments.**

## I. INTRODUCTION

The recent research work related to automatic service composition in pervasive environments has gained much maturity, and together with advancements in various technologies, this concept has reached a level where the life of an ordinary user is completely automatized. Emphasis has been on the automatic selection of services for users, without their intrusion, in the pervasive environment. In most cases, such an approach considers an abstract user task on the user device, which leads to automatic selection of services across various devices in the environment, according to the current context.

For example, consider a video player application that provides the functionality of displaying video to the user. The user is also able to control the playback of the application. The application is represented by an assembly of abstract components, which describe only the services required or provided by the application namely, controlling, decoding and displaying video. The application has to be mapped to the concrete components to achieve its realization.

The complexities involved in designing and realizing such applications have been identified and addressed by many previous approaches [2] [5] [14]. Mostly, they consider both the functional and non-functional aspects of the application. For example, in one of our previous work, we have presented a mapping algorithm that maps an abstract application to concrete one considering functional aspects of the application —interfaces and message interactions— as well as non-functional aspects like user preferences, devices' capabilities and network protocol heterogeneity to ensure the mapping of services to the components [14].

While the existing approaches may assume that a mapping from abstract to concrete application can be done effortlessly, many problems may rise that prevent it to be achieved successfully. As a first case, consider the heterogeneity of interfaces of different components or devices, heterogeneity of interaction messages, etc. Furthermore, applications in pervasive environments are challenged by the dynamism of their execution environment due to, e.g., user and device mobility, which make them subjects to unforeseen failures. An automatic adaptation is to remap the abstract description of application to the concrete level. The remapping corresponds to the reselection of new concrete components to replace some others.

Both of these problems imply mismatches between abstract and concrete levels that may occur either at the time of mapping during initialization or even after the application has been executed. Thus, adapting abstract applications represents a crucial need to be considered in order to ensure their mapping to the concrete level and their execution.

The problem of adapting component-based models has been extensively studied in different contexts, notably in component-based applications. In the literature, we distinguish many relevant adaptive approaches that propose parametric or compositional mechanisms to adapt applications in pervasive environment (e.g., [2] [18] [19]). Parameterization techniques aim at adjusting internal or global parameters in order to respond to changes in the environment. Compositional adaptation is classified into structural and behavioural adaptations [13]. We mean by behavioural adaptation the modification of the functional behaviour of application in response to changes in its execution environment. However, structural adaptation allows the restructuring of the application by adding or removing software entities with respect to its functional logic.

In this article, we propose a dynamic structural adaptation

 ISBN: 978-1-61208-109-0 

approach based on adaptation patterns to provide solutions to the detected mismatches between abstract application and the concrete level. Adaptation patterns are injected into the abstract application to provide extra-functional services allowing its mapping and its execution. Thus, the abstract application is transformed to another one that ensures its mapping and its execution. To facilitate the description of the adaptation patterns, we define a generic adapter template that encapsulates the main features of an adapter.

We are not interested in describing the detection of adaptation context; rather, our objectives are: 1) to define an adapter template and to give examples of adaptation patterns based on this template and 2) to show how adaptation patterns are used to adapt the abstract application and hence the concrete one.

The remainder of this paper is organized as follows. Section II mentions and classifies examples of factors triggering the adaptation of abstract application. Section III describes the principle of our structural adaptation approach illustrated by examples of adaptation patterns. In Section IV, we present an example scenario through which we show how patterns are used to adapt dynamically an application. In Section V, we present some implementation details. Section VI provides an overview of existing related approaches as well as their limitations. Finally, Section VII concludes the article with an overview of our future work.

## II. Adaptation Contexts

A generalized notion of context has been proposed in [1] as *any information that can be used to characterize the situation of an entity (person, location, object, etc.)*. We consider adaptation context as any piece of information that may trigger the adaptation of the application. We are interested in contexts that represent the mismatches between abstract and concrete levels. These mismatches imply that the current abstract description could not be realized in the given context, or in the new context, if it has changed.

We classify the factors triggering the adaptation of the abstract description into three categories: 1) factors related to the software characteristics of devices, 2) factors associated with the network characteristics of devices and 3) factors related to the hardware characteristics of devices. Our intent through the given classification is to cover the different aspects related to software, network or hardware sides that may trigger the adaptation of the abstract application.

The software factors are related to the software components of the application through which it describe its functionalities. A mismatch related to software factors includes interface mismatches due to different syntactic representation, differences of the interaction messages supported by components, requiring component wrappers to hide implementation heterogeneity, or adding proxy components to control access, etc.

In hardware factors, we consider the factors associated with the hardware characteristics of devices. A change in hardware configuration may lead to change in the context, requiring adaptation. Such changes may imply improvement or decline in the device capacities, e.g., high CPU usage, reduced capacity of memory storage, attachment of new hardware components, etc.

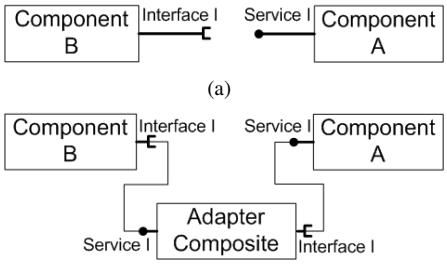



Fig. 1. Transforming abstract application using Adapter composite

In addition to the software and hardware factors, there are several challenges related to the network characteristics of a device that may require the adaptation of the application, e.g., use of different interfaces of network connections such as changing to Bluetooth from Wi-Fi if the latter disappears, the fluctuation in the quality of network signals, etc.

The example factors are some of recurrent causes that are responsible for the mismatches between abstract and concrete levels. There may be other factors that trigger the adaptation of the abstract description of the application to support the changes of the environment (e.g., user preferences); however, we do not deal with them in this article.

In the next section, we detail the proposed adaptation approach to overcome the detected mismatches that are triggered by some factors related to software, hardware or network characteristics of devices.

## III. Adaptation Patterns

### A. Principle of our Approach

To overcome the captured mismatches between abstract and concrete levels, we propose to transform an abstract application to another one that ensures its mapping and its execution. The transformation consists of injecting extra-functional adapters into the abstract application.

As shown in figure 1, an adapter composite is injected to adapt the interaction between components A and B. The adapter composite requires the service I of the component A and exposes a service implementing the interface I. This provided service will be used by the component B, since it corresponds to its required service. Thus, the abstract application is transformed by adding extra-functional behaviour to achieve its mapping or its execution.

We note that adapters are identified at execution time without the assistance of the designer. This can be done by using a set of adaptation policies defined at the design time. These policies can then relate the adaptation pattern with the execution context. However, in the present work, we do not tackle the identification of the used adapters that remains an objective for the future work.

 ISBN: 978-1-61208-109-0 

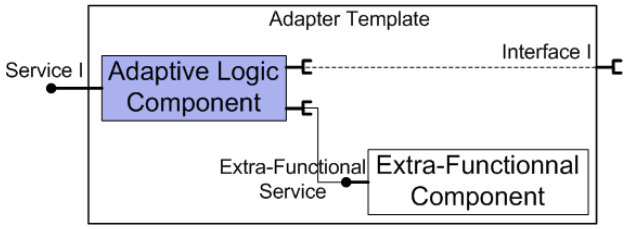



Fig. 2. Generic adapter template

### B. Adapter Template

As the basis for our approach we propose to use adaptation patterns as adapter composites to provide solutions for the detected mismatches between abstract and concrete levels. Patterns have component-based descriptions that encapsulate extra-functional services for the adaptation of abstract applications.

We have defined an adapter template to be used for the description of adaptation patterns. Figure 2 shows the description of the adapter template, which consists of an ”adaptive logic” component and an ”extra-functional” one.

The extra-functional component provides transformation services allowing, e.g., encryption, compression, etc. The adaptation pattern provides an abstract description for the extra-functional component to be mapped following the matching algorithm [14]. Thus, the extra-functional component is identified dynamically without the assistance of designer. Its offered service will be used by an adaptive logic component which acts as an intermediate between the abstract and the extra-functional components.

For this, the adaptive logic component is considered as the main element of the adapter template. And each adaptation pattern should contain at least the adaptive logic component, if it does not require any transformation service to overcome the mismatch (see Section IV). The implementation of the adaptive logic component is generated, since it depends to the business interfaces of abstract components. As shown in figure 2, the adaptive logic component requires the offered service of the extra-functional component and a service implementing the interface I, which corresponds to the service I of the component A. Otherwise, the component provides a service implementing the interface I as the required service of the component B.

Using this specific structure of the adapter template has an advantage, on the one hand, to separate the adaptive logic from the functional logic of the application. Thus, it is possible to modify the adaptive logic with respect to the components' descriptions. On the other hand, it allows providing an abstract description for the adaptation actions, corresponding to extra-functional components, to be mapped following the matching algorithm. Thereby, the separation between adaptive logic and extra-functional components facilitates the generation of the adaptive logic that plays an intermediary role between abstract components and extra-functional ones.

Due to space limitation, in this section, we only discuss three adaptation patterns that are defined based on our introduced adapter template.

### C. Example Adapter Patterns

*1) Compressor Pattern:* A compressor pattern is used to handle a mismatch between an abstract application and a concrete level triggered by a network factor, which is the fluctuation of the network signal used by devices. Thus, if the signal strength is high, bigger data can be sent over network. However, if the signal strength is weak; data should be compressed for a quick transfer.

The compressor pattern has the given description by figure 3. It consists of an adaptive compression component and a compressor one. The abstract compressor component will be mapped to the concrete level to identify the corresponding concrete component that implements the described compression interface. However, the adaptive compression component will be generated to implement the interface I as required by the component B. Its implementation contains an invocation of the provided service of the compressor component in addition to the service I.

*2) Decompressor Pattern:* A decompressor pattern is used whenever a compressor pattern is handled by a device to overcome the weakness of the network signal. Indeed, within a compressor pattern, we require a decompressor pattern to decompress the data before using it by the component A.

The decompressor pattern is described following the adapter template as shown in figure 3. It contains an adaptive decompression component and a decompressor one. While the decompressor component is mapped to the concrete level, the adaptive decompression component is generated to provide a service I. The implementation of the adaptive logic component consists on applying a decompression algorithm supported by the service of the decompressor component before invoking the service I provided by the component A.

*3) Proxy Pattern:* The proxy pattern, as defined by Gamma et al. [9], provides solutions to problems related to inaccessible software entities. Thus, it would be useful to overcome the network factor related to different interfaces of connections as detailed in the next section IV.

Figure 4 gives a component-based description of the proxy pattern following the adapter template. As it can be seen, the proxy pattern represents a specific case of the adapter template. It contains only a proxy component representing the adaptive logic component that forwards the call of the service I to the component A.

## IV. Example Scenario Using Adaptation Patterns

Referring back to the video player application described in the introductory section, figure 5 shows an abstract description of the Video player application that consists of three components: a *VideoDecoder* component, a *DisplayVideo* component and a *Controller* Component. The *Controller* component sends a command to the *VideoDecoder* component to decode a stored video. The *VideoDecoder* component decodes a video into appropriate format. Once the video is decoded, it is passed

 ISBN: 978-1-61208-109-0 

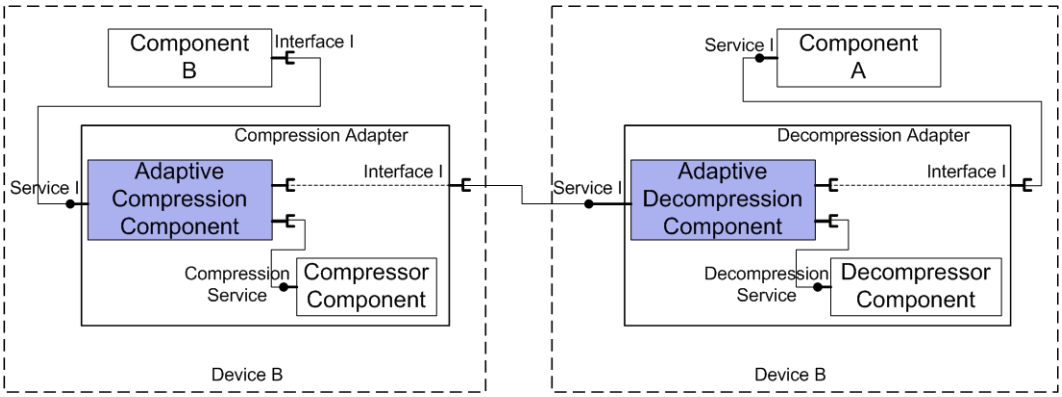



Fig. 3. Compressor and decompressor adaptation patterns

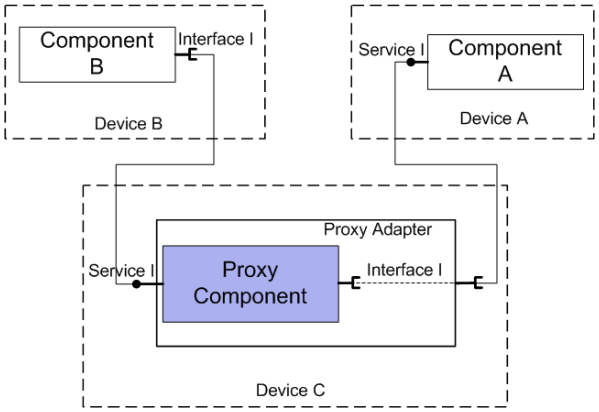



Fig. 4. Proxy adaptation pattern

to the *DisplayVideo* component to play it. This is done using the service provided by the *DisplayVideo* component through an appropriate programming interface.

The given execution environment consists of a Smartphone (SP) device that supports a Wi-Fi interface connection and contains the *Controller* and the *VideoDecoder* components. There is also a flat-screen (FS) device that uses a Bluetooth interface connection and maintains the *DisplayVideo* component. A laptop (LP) device is available in the execution environment and supports both interfaces of connection (i.e., Bluetooth and Wi-Fi). However, LP does not provide any services described in the video player application. Thus, only the smartphone and the screen devices would be used to support the application mapping.

However, both of these devices cannot interact between them, since they support different connection interfaces. Thus, there is a mismatch between the given abstract description of the video player application and the concrete level because of a network factor that causes the failure of the mapping. Therefore, the Video Player application should be adapted to achieve its mapping.

To overcome the heterogeneity of connection interfaces, we propose to use a proxy adaptation pattern to create a representative for the *DisplayVideo* component. As described in III-C3, the composite of the proxy pattern contains only a *DisplayVideoProxy* component representing the adaptive logic. It requires the *DisplayVideoService* to forward the call of the *VideoDecoder* component to the *DisplayVideo* component. The proxy pattern is generated on the laptop device since this latter supports Wi-Fi and Bluetooth connections. As a result, the application is transformed, as shown in figure 6, to contain the DisplayVideoProxy component in addition to its own components.

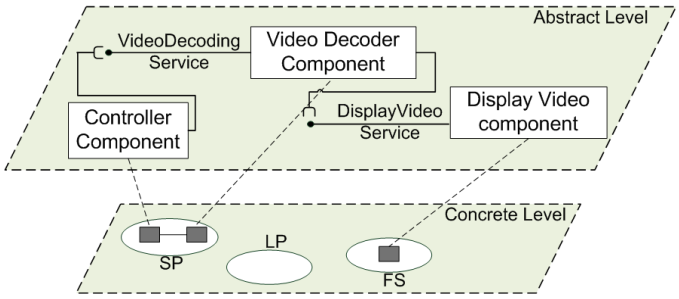



Fig. 5. Video Player application

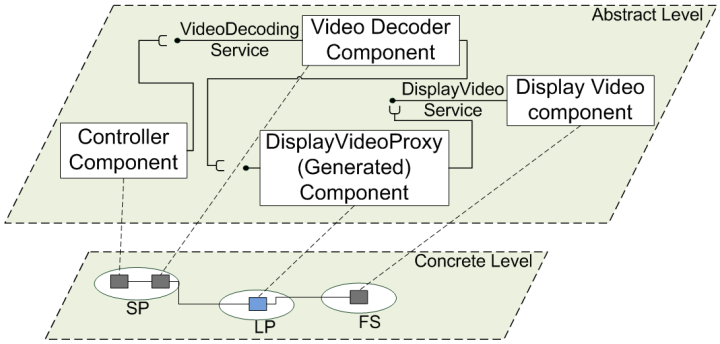



Fig. 6. Adaptation of the Video Player application

## V. IMPLEMENTATION

We have implemented, as a proof of concept, a java prototype of a service allowing the automatic generation of pattern's components following the description of adapter

 ISBN: 978-1-61208-109-0 

template. SCA (Service Component Architecture) [15] is used to describe components since it provides the ability to write applications abstractly and then map them to the concrete components. One of the main features of the SCA component model is that it is independent of particular technology, protocol, and implementation.

The open source Javassist [1] (JAVA programming ASSISTant) library is used to generate the byte code of an adaptive logic component that implements a specified interface I. For this purpose, we use the `java.lang.reflect` package to obtain reflective information about methods of the interface I (i.e., names, parameters, etc).

Similarly, if the adaptive logic component requires an extra-functional component, the interface of this latter is introspected to be used for the implementation of the adaptive logic component.

In case of our example Video Player application, the interconnectivity between devices is represented by a graph [14]. For the given scenario, there will be no connection between smartphone and flat-screen devices in the graph, representing an interface mismatch. However, connections between smartphone and laptop, and between laptop and flat-screen are detected in the graph, which indicates a proxy pattern should be generated in the laptop device to overcome interface mismatching. The pattern consists of the DisplayVideoProxy component that implements the introspected interface of the DisplayVideo service and whose byte code is generated dynamically using the Javassit library.

## VI. Related Work

In this section, we detail some of the existing structural adaptation approaches as well as their limitations.

In [17], Sadjadi et al. propose a general programming model called transparent shaping that supports the design and the development of adaptable programs from existing programs. In transparent shaping, an application is augmented with hooks that intercept and redirect interaction to adaptive code. Integration of the adaptive code is a two-phase process, in the first phase; the application is transformed into a new adapt-ready application by weaving hooks into the application code. In the second phase, usually executed at run-time, an adaptation infrastructure is inserted dynamically using these hooks.

Compared to our approach, the major drawback of their approach is that the identification of the hooks is done at development time. Besides, the program should be in ready state, using a static transformation, to support the insertion and the removal of adaptive code at runtime. However, in our case, we provide a dynamic adaptation for applications at init time as well as at runtime without any transformation in advance.

[16][8] present an AO-ADL (Aspect Oriented-Architecture Description Language) language. AO-ADL considers that components model either crosscutting (named aspectual component) or non-crosscutting behaviour (named base component) exhibiting a symmetric decomposition model. Towards this challenge, they propose to use aspectual connectors that provide support to describe and to weave aspectual components to the regular ones. These connectors are described following a template connector.

However, the instantiation of the connector template is done at design time, which limits the possibility to extend applications with new aspects. Furthermore, they consider connector template per aspect. However, it is not possible to define as many connector templates as possible aspects.

In another related work, AO-OpenCom [10] [18] that is an Aspect Oriented Middleware platform, provides a dynamic reconfiguration of distributed aspects. The platform builds on the OpenCom component model [7] and the distributed component framework. Aspect composition in AO-OpenCom employs components to play the role of aspects. Aspects are composed at runtime using so-called interceptor-connectors that support the dynamic insertion of aspect components. Aspects can be added to a system at run-time using an aspect-oriented MOP that allows a fine-grained introspection and an adaptation of cross-cutting behaviour.

While their approach is quite general, we can identify two limitations compared to our proposed approach. First, aspects are used only to adapt OpenCom-based application at runtime. Second, the join points are already defined by designer, which limits the number of adaptations.

[19] presents a WComp middleware that uses the concept assembly of aspects (AA) to modify the internal component assembly of event-based web services [11]. Aspects of assembly are pieces of information describing how an assembly of components will be structurally modified. In such mechanisms, aspects are selected either by the user or by a self-adaptive process and composed by a weaver with logical merging rules. The result of the weaver is then projected in terms of pure elementary modifications of components assemblies.

However, the weaving and the validation rules are predefined by the developer at design time, which limits the number of adaptation. Moreover, they do not take into account to the adaptation of applications at initialization.

In [3], Becker et al. propose an adaptation model which is built upon a classification of component interfaces' mismatches. To cover these mismatches, they identify a number of patterns to be used for eliminating the interfaces' mismatches. They classify the adaptability pattern into two categories; functional adaptation pattern to bridge the functional component incompatibilities and extra functional adaptation pattern to increase a single or several quality attributes of the component being adapted.

However, these adaptation patterns are identified at design time to achieve the adaptation of applications at runtime. Besides, they limit the adaptation contexts to the mismatches between components' interfaces. Thus, they do not take into account the network and hardware factors related to characteristics of devices.

Other related work in this area [12] [6] have also investigated matching of Web service interfaces by providing a classification of common mismatches between service interfaces

[1] http://www.csg.is.titech.ac.jp/~chiba/javassist/

and business protocols, and introducing mismatch patterns. These patterns are used to formalize the recurring problems related to the interactions between services. The mismatch patterns include a template of adaptation logic that resolves the detected mismatch. Developers can instantiate the proposed template to develop adapters. For this purpose, they have to specify the different transformation functions.

We can identify two important limitations compared to our approach. First, the mismatch patterns are limited to the interfaces and protocols mismatches. Thus, they do not consider the other different software, network and hardware factors. Second, the specification of adapters is done with the help of developers. However, in our approach, we are able to specify dynamically the different components of a used pattern; by generating the implementation of its adaptive logic component and mapping the extra-functional one following our matching algorithm [14].

## VII. Conclusion and Future work

In this article, we have proposed an approach for dynamic structural adaptation of abstract applications. We limit the adaptation contexts on factors implying mismatches between abstract application and the concrete level. These factors are related to software, hardware and network characteristics of devices and they are detected at init time or during the execution of the application.

To overcome these mismatches, we propose to use adaptation patterns that provide extra-functional behaviour to the abstract applications. The adaptation patterns are described following our generic adapter template, which consists of adaptive logic and extra-functional components. While the adaptive logic component has a generated implementation, the extra-functional component is identified following a mapping process in [14].

Using this approach allows to separate the extra-functional logic from the business one, and hence, to add or remove adaptation patterns dynamically from the abstract application whenever there is a need.

We are looking forward to integrate the implemented prototype in our SCA platform [4] and the Newton SCA runtime[2]. And in the near future, we will tackle the identification of adaptation patterns that are used to support the adaptation of abstract applications.

## VIII. ACKNOWLEDGMENTS

This work is partially supported by French ANR through Seamless (SEamless and Adaptive Services over MultipLe AccEsS NetworkS) project number 07TCOM018.

## References

[1] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, pages 304–307, 1999.

[2] Christian Becker, Marcus Handte, Gregor Schiele, and Kurt Rothermel. Pcom - a component system for pervasive computing. In *PERCOM '04: Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, page 67, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Steffen Becker, Antonio Brogi, Ian Gorton, Sven Overhage, Alexander Romanovsky, and Massimo Tivoli. Towards an engineering approach to component adaptation. In *Architecting Systems with Trustworthy Components*, pages 193–215, 2004.

[4] Djamel Belaïd, Hamid Mukhtar, Alain Ozanne, and Samir Tata. Dynamic component selection for sca applications. In *I3E*, pages 272–286, 2009.

[5] Sonia Ben Mokhtar, Nikolaos Georgantas, and Valérie Issarny. Cocoa: Conversation-based service composition in pervasive computing environments with qos support. *J. Syst. Softw.*, pages 1941–1955, 2007.

[6] Boualem Benatallah, Fabio Casati, Daniela Grigori, H. R. Motahari Nezhad, and Farouk Toumani. Developing adapters for web services integration. In *In Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE), Porto,Portugal*, pages 415–429. Springer Verlag, 2005.

[7] Geoff Coulson, Gordon Blair, Paul Grace, Ackbar Joolia, Kevin Lee, and Jo Ueyama. A component model for building systems software. In *IASTED Software Engineering and Applications (SEA04)*, 2004.

[8] Lidia Fuentes, Nadia Gámez, Mónica Pinto, and Juan A. Valenzuela. Using connectors to model crosscutting influences in software architectures. In *ECSA*, pages 292–295, 2007.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[10] Paul Grace, Bert Lagaisse, Eddy Truyen, and Wouter Joosen. A reflective framework for fine-grained adaptation of aspect-oriented compositions. In *SC'08: Proceedings of the 7th international conference on Software composition*, pages 215–230. Springer-Verlag, 2008.

[11] Vincent Hourdin, Jean-Yves Tigli, Stéphane Lavirotte, Gaëtan Rey, and Michel Riveill. Slca, composite services for ubiquitous computing. In *Mobility '08: Proceedings of the International Conference on Mobile Technology, Applications, and Systems*, pages 1–8. ACM, 2008.

[12] Woralak Kongdenfha, Hamid Reza Motahari-Nezhad, Boualem Benatallah, Fabio Casati, and Regis Saint-Paul. Mismatch patterns and adaptation aspects: A foundation for rapid development of web service adapters. *IEEE Transactions on Services Computing*, pages 94–107, 2009.

[13] Philip. K. McKinley, Seyed Masoud Sadjadi, Eric P. Kasten, and Betty H. C. Cheng. A taxonomy of compositional adaptation. Technical report, 2004.

[14] Hamid Mukhtar, Djamel Belaïd, and Guy Bernard. A graph-based approach for ad hoc task composition considering user preferences and device capabilities. In *Workshop on Service Discovery and Composition in Ubiquitous and Pervasive Environments*, New Orleans, LA, USA, dec 2007.

[15] Open SOA Collaboration. Service component architecture (sca): Sca assembly model v1.00 specifications. http://www.osoa.org/, 2007.

[16] Mónica Pinto and Lidia Fuentes. Ao-adl: an adl for describing aspect-oriented architectures. In *Proceedings of the 10th international conference on Early aspects*, pages 94–114. Springer-Verlag, 2007.

[17] S. Masoud Sadjadi, Philip K. McKinley, and Betty H. C. Cheng. Transparent shaping of existing software to support pervasive and autonomic computing. In *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, pages 1–7. ACM, 2005.

[18] Bholanathsingh Surajbali, Geoff Coulson, Phil Greenwood, and Paul Grace. Augmenting reflective middleware with an aspect orientation support layer. In *ARM '07: Proceedings of the 6th international workshop on Adaptive and reflective middleware*, pages 1–6. ACM, 2007.

[19] Jean-Yves Tigli, Stephane Lavirotte, Gaëtan Rey, Vincent Hourdin, Daniel Cheung-Foo-Wo, Eric Callegari, and Michel Riveill. Wcomp middleware for ubiquitous computing: Aspects and composite event-based web services. *Annales des Télécommunications*, pages 197–214, 2009.

[2] http://newton.codecauldron.org/site/concept/ComponentModel.html

# The Use of "Canaries" for Adaptive Health Management of Electronic Systems

Abhijit Dasgupta, Ravi Doraiswami, Michael Azarian, Michael Osterman, Sony Mathew, and Michael Pecht
Center for Advanced Life Cycle Engineering (CALCE), University of Maryland,
College Park, MD 20742, United States

*Abstract* - **Reliability concerns in state-of-the-art electronic systems have led researchers and engineers to develop innovative real-time prognostics and adaptive health management methods to assure desired availability. Prognostics techniques described here use a novel concept of *canaries*, along with data analysis, failure mechanism models and integrated fusion techniques to determine the remaining useful life of a system. A *canary* is a device that provides data to generate early warning of functional degradation and impending functional failure. Three types of *canaries* are discussed. Expendable *canaries* experience accelerated degradation (compared to functional degradation) by design, so that early warning of impending failure can be generated. Sensory *canaries* provide early warning by observing nonfunctional manifestation of functional degradation. Conjugate-stress *canaries* provide measurement of life-cycle stress history so that failure models can be used to estimate consumed life and remaining life. This paper focuses on expendable canaries, in particular, and provides three examples to illustrate the underlying design concepts.**

***Keywords – canary; electronic systems; failure mechanisms; precursors; prognostics; reliability; remaining useful life.***

## I. INTRODUCTION

Cost effective methods for assuring high availability and safety are a key need in many critical electronics-rich systems, such as in medical devices, military products, aviation controllers and avionics, information management systems and in energy generation and distribution systems, especially in nuclear power plants [1]. Functional failure of such critical products during performance in the field can have catastrophic and costly consequences. The prevention of failures in cutting-edge electronics is very challenging because of the extremely high density and functionality, small length-scales, multitude of competing multi-physics degradation mechanisms and failure mechanisms, rapid turnover of technology and highly diverse global supply chain.

Current predictive methods in the industry for predicting failures, based on reliability handbooks, have been shown to be very inadequate. Some of the key causes for prediction inaccuracies are model inaccuracy and uncertainties in the inputs to the models. The most important input is the life cycle loading history and uncertainty arises in this input users. Another important source of uncertainty is in input model constants that represent material behavior, because of the wide variability in defect levels and manufacturing tolerances in specimen populations. These variabilities are difficult to quantify *apriori* in new emerging technologies, making it difficult to make accurate proactive failure predictions. Corrective and preventive maintenance can, to some extent, mitigate failure risks [2]. However, mandatory maintenance on an inflexible predetermined schedule is not always the most cost-effective way to ensure availability and safety. Prognostics and health management (PHM) methodology is based on monitoring the performance of a system *in-situ* in its actual life-cycle conditions to identify early signs of degradation, providing advance warnings for future failures, and implementing timely corrective action to mitigate the failure risk. As an important aspect of PHM, the technology of embedded *canaries* has attracted more and more attention from industry due to their intrinsic capability to provide early warning of host degradation and impending failure, under actual life-cycle conditions.

The use of the term *canary* in PHM is derived from the historic concept of using canary birds in coal mines in order to detect the presence of hazardous gases. Because canaries are more sensitive to hazardous gases than humans, the death or sickening of the canaries provided early warning to miners to take corrective action and exit the mine-shaft. In this paper, we use the word *canary* to refer to a family of technologies that are embedded in functional electronic systems, to give early warning of functional degradation, and to provide estimates of the remaining useful life of the host system. In a classical context, the word *canary* was used to refer to a device that changes its functional characteristics ahead of similar functional parameters that indicate system performance degradation when the product is subjected to life cycle stress conditions. If the acceleration factors are known, their time-to-failure estimates can be used to quantify the remaining useful life of the system. The early warning provided opportunities to implement timely risk-mitigation actions and avert danger. The use of *canaries* is an integral part of the prognostics approach described in this paper, and the concept is generalized and extended well beyond the classical definition, as discussed later in this paper.

*Canaries* have been applied to several system-level applications to predict failures and the resulting changes in the performance of systems. In system-level applications

 ISBN: 978-1-61208-109-0 

they have had a significant impact in failure prediction by providing advance warning about the initiation and growth of defects and their correlation to system parameters. Researchers have been working on applying *canaries* to the prognostics of both components and systems. Han et al. [3] proposed a concept of developing a "*canary*-containing" packet that can be attached externally to weapon casings. In this process, the *canaries* received environmental loading identical to what the casings experienced. The *canary* material was configured to deteriorate at a faster rate than the system's parameters. Energetic materials that have a faster rate of initiation of failure mechanisms compared to semiconductors materials have been applied to semiconductor systems to evaluate their failures and manage their health by applying pre-calibrated circuits to the host chip (Mishra et al. [4]).

Ridgetop Group has developed sentinels for advance warnings of device failures [5]. Wang et al. [6] demonstrated a 90nm 128Kb SRAM test chip on which the *canary* cells track changes in temperature and data retention voltage, which is the minimal value of positive supply voltage ($V_{DD}$). These *canary* cells ensure a reliable function of the host chip by protecting core cells in a closed loop $V_{DD}$ scaling system. Calhoun et al. [7] proposed a closed-loop approach using *canary* flip-flops to enable power savings of over 40 times in a 0.13-um, dual-test chip. Anderson et al. [8] used low cycle fatigue solder joints and corrosion-susceptible circuits on the host printed circuit board as *canaries* and identified prospective failure mechanisms. Goodman et al. [9] used a prognostic cell to monitor time-dependent dielectric breakdown (TDDB) of the metal oxide semiconductor field-effect transistor (MOSFET) on integrated circuits.

In this paper, we present several examples of *canaries* for electronic systems, but first provide the context of the prognostics framework that *canaries* are a part of. The prognostics framework to monitor and predict the remaining life of systems is reviewed in Section II. A detailed review of *canary* design and application is presented in Section III. Examples of *canaries* and their applications are provided in Section IV. In Section V conclusions are provided, including future functional goals for *canaries*.

## II. PROGNOSTICS METHODOLOGY

To improve the availability of a system, we need to ensure that the system can perform as intended (i.e., without failure and within specified performance limits) for a specified period of time, in its life-cycle environment [10]; within a specified confidence level. Unfortunately, traditional reliability assessment methods using Telcordia [11], PRISM [12], and FIDES [13] fall short in performance, since they fail to provide accurate failure predictions, which can result in poor design and logistic decisions [14] [15]. The central idea in prognostics is to assess the reliability in real-time, under its actual application conditions, so that timely corrective action can be implemented and availability can be improved. The general prognostics methodology is shown in Figure 1 [16]. Prognostics techniques combine sensing, recording, and interpretation of environmental, operational, functional and relevant non-functional parameters, to determine failure precursors that are indicative of a system's health. A failure precursor is a data event or trend that signifies impending failure. A precursor indication is usually a change in a measurable variable that can be associated with system degradation and subsequent failure. For example, a shift in the output voltage of a power supply might suggest impending failure due to a damaged feedback regulator and damaged opto-isolator circuitry. Pecht et al. [18] presented a guideline for selecting measurable parameters that can be used as failure precursors for electronic products.

The first step in Prognostics involves a virtual life assessment where design data, expected life-cycle conditions, the results of failure modes, mechanisms, and effects analysis (FMMEA), and Physics of Failure (PoF) models are the inputs. Based on the virtual life assessment, it is possible to prioritize the critical failure modes and failure mechanisms. The existing functional data, bus monitor data, BIT, IETM, *canary* data, maintenance and inspection records can be used to identify potential the abnormal conditions and parameters. Based on this information, PHM engineers select the functional parameters to be monitored for health assessment and relevant sensor locations as well as relevant *canary* designs. Based on the collected functional and *canary* data, the health status of the products can be assessed [10]. The different approaches to prognostics are highlighted in yellow boxes in Figure 1. Three current approaches for Prognostics include: (A) Data-driven Prognostics; (B) Model-based Prognostics; and (C) Prognostics based on fusion of model and data.

*A. Data-driven Prognostics:* This approach is based on trend analysis of the precursor data from the functional system and the *canaries*. The monitored data is analyzed to see if there is a significant change in the previous trends observed with this data. Trends are detected by extracting key features which are then classified as recurring anomalies or transient soft events, using machine learning methods. Time series techniques and logic-reasoning techniques are used for state estimation, to diagnose the underlying cause and source of degradation. Since the data is highly multi-dimensional, *canary* data can help to deconvolve them for insights about dominant failure mechanisms. Forecasting methods and dynamic state tracking methods are used to extrapolate this data and provide predictions of when failure is likely to occur and hence provide an estimate of the remaining useful life of the system. Techniques such as parametric curve fit, expert system, neural networks, and Bayesian inference are often used for this task [19]-[28]. Methods, such as FMMEA [29], are used to determine parameters that need to be monitored.

*B. Model-based Prognostics:* Life-cycle loads (thermal, mechanical, chemical, electrical, and so on) on a product can

 ISBN: 978-1-61208-109-0 

arise from the manufacturing, shipment, storage, handling, and operating and non-operating conditions, and may lead to performance or physical degradation of the product and reduce its service life [16]. Physics-of-Failure (PoF) models provide quantitative stress-damage relationships, based on the root causes of the failure of *canaries* and functional components and provide insights about how fast the system degrades and when failure can be expected, when used in conjunction with the precursor data from the functional systems and from the *canaries*. PoF models are typically based on fundamental physics and may take a relatively simple approach at the continuum length-scale or a more complex *ab-initio* approach at the molecular and atomistic length scales [17].

*C. Fusion Prognostics* This approach provides an intelligent combination of the model-based and data-driven methods to improve the accuracy and speed of diagnostics and prognostics. Data-driven approaches have the drawback that they require vast amounts of training data and are difficult to extrapolate to new use conditions. Methods based purely on PoF models have the drawback that real-time calculation of failure models for a complex system may be computationally very cumbersome. Fusion between the two provides valuable synergy and also provides the ability to discriminate between real failure trends vs. transient false-positives.

Self-checking circuitry and built-in-tests (BITS) can also be incorporated, in addition to *canaries*, to sense abnormal changes in system functions and to activate autonomous reconfiguration, to compensate for the malfunction [30].

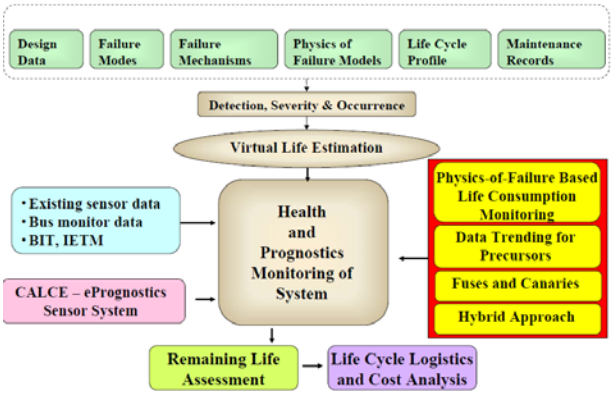



Figure 1. Prognostics methodology.

## III. DESIGN OF CANARIES

*Canaries* are embedded in products and are used to identify changes in failure mechanisms as a precursor to predict the future effects of similar mechanisms on system parameter changes. *Canaries* are integrated into components or devices during the system design phase. These embedded *canary* devices (also called prognostic cells) are non-critical elements in overall system performance [31].

Figure 2 shows the failure probability density distributions for functional products and corresponding *canaries*, showing the concept of remaining useful life (RUL) or 'prognostic distance.'

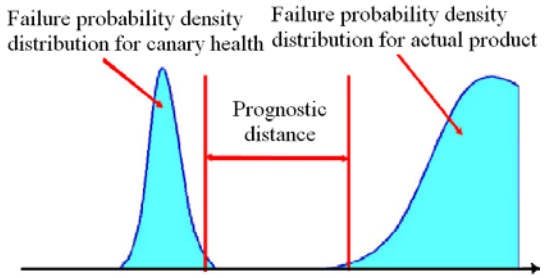



Figure 2. Failure probability density distributions for *canaries* and actual products, showing prognostic distance or remaining useful life (RUL).

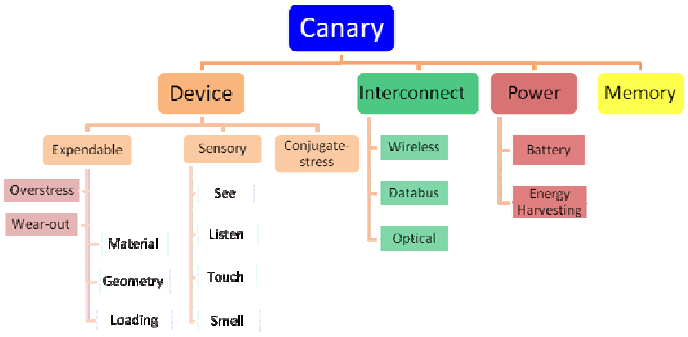



Figure 3. Classification of *canaries.*

The term c*anaries*, as used in an extended generalized context in this paper, refers to three main types of devices (see Figure 3): expendable *canaries*, sensory *canaries*, and conjugate-stress *canaries*.

*A. Expendable canaries:* They are based on controlled acceleration of failure precursor signatures, using error-seeded, sacrificial, nonfunctional elements (*canaries*). The error-seeding in *canaries* includes three inter-related techniques that will be used individually or synergistically to enhance the degradation rates in the *canaries*: geometry error-seeding, material error-seeding, load error-seeding. Section IV will discuss these three error-seeding methods in detail. Based on their dominant failure mechanisms, expendable *canaries* can be further categorized into overstress *canaries* and wear-out *canaries*. Overstress *canaries* come into play when loaded stress exceeds its strength. Some of the overstress failures are dielectric breakdown, electrostatic discharge (ESD), interconnect fracture, pin buckling. Wear-out failure, which is caused by gradual increase of cumulative damage, includes electromigration, fatigue, Tin whisker growth, interconnect corrosion, time dependent dielectric breakdown caused by tunneling mechanisms, etc.

*B. Sensory canaries:* Inspired by biological system, these *canaries* can detect non-traditional signatures of system degradations. They will be able to "look," "listen," "smell," and "feel" for signs of degradation and impending failure

 ISBN: 978-1-61208-109-0 

just as an animal does. By collecting and analyzing these signs, early warnings of failure can be achieved. Examples include: infrared *canaries* to "look" for degradation in microprocessors based on changes in the thermal dissipation; Impedance spectroscopy and time domain reflectometry to "listen" for defects in signal traces and wiring harnesses. Micro Electro-mechanical System (MEMS)-based chemical *canaries* to "smell" for outgassing products; Piezoelectric or piezoresistive *canaries* to "touch" for signs of delamination, which is a common failure mechanism of laminated material subjected to repeated cyclic stresses or impact, with significant loss of mechanical toughness.

*C. Conjugate-stress canaries:* These *canaries* can provide model-based fusion prognostic assessments of remaining useful life based on simultaneous identification of conjugate-stress pairs, which is a novel concept because most conventional detectors provide a single stress metric, rather than a conjugate pair. For example, a conventional stress monitoring *canary* such as a thermocouple can measure only temperature. In many cases, it cannot provide sufficient information to health condition. However, the conjugate-stress approach will simultaneously measure temperature and heat flux at the same site by using a pair of collocated detectors.

These three classes of *canaries* will provide unprecedented levels of synergistic prognostic and health information of systems to minimize uncertainties in remaining useful life assessment. In this paper we discuss only the first type, viz. expendable *canaries*. Discussion of the two remaining types of *canaries* is deferred to a future paper

Expendable *canaries* can be designed based on controlled error-seeding, which provides controlled acceleration of degradation rates and failure precursor signatures. Technically, an expendable *canary* can be any device that wears out faster than the actual product under the same environmental and operational loading conditions. An example is a dummy filament in a light bulb, which is designed to age faster than functional filament can be used as an expendable *canary* to provide early warning of impending failure of the bulb. The methods to make the filament fail faster than normal one can be either scaling its physical dimension, or subjecting it to higher current stress, or utilizing material of less susceptibility to melting. We called these methods geometry error-seeding, material error-seeding, load error-seeding.

*A. Material error-seeding*

Material error-seeding takes advantage of the greater sensitivity of the materials to failure modes (physical or chemical),in *canaries* compared to that in the functional system so that degradation and failure of *canaries* can be used as an early warning of the degradation and future failure of of the functional host, before experiencing catastrophic loss. Since there are various properties a material has, many types of *canaries*, in theory, can be designed. These properties include, but are not limited to, the following: mechanical, thermal, electrical, magnetic, optical and photonic, chemical, biological, reaction to gases, and reaction to humidity. Each category has quite a few material properties; for example, the mechanical properties include Young's modulus, specific modulus, tensile and compressive strength, ductility, Poisson's ratio, etc. Thermal properties include thermal conductivity, thermal diffusivity, coefficient of thermal expansion, specific heat, glass transition temperature, melting point, Curie point, etc. Electrical properties include electrical conductivity, permittivity, dielectric constant, dielectric strength, Seebeck coefficient, etc. The selection of material properties is determined by the application of *canaries* and other concerns, such as the effectiveness of providing early warning, ease of fabrication, and low cost.

*B. Geometry error-seeding canaries*

This kind of *canary* is designed to change in shape, or geometry in response to changes in failure mechanisms induced by a stimulus. Ridgetop Group's commercialized prognostic cells can provide an early-warning sentinel for device failures [5]. Their prognostic cells are available for 0.35, 0.25, and 0.18 micron CMOS processors. The time to failure of these prognostic cells depends on the stress on the circuit, including voltage, current, temperature, humidity, and radiation. The earlier failure of these prognostic cells is achieved by the controlled shrinking of the cross-sectional area of the circuits inside the cells. With the same amount of current passing through host circuits and *canary* circuits, the *canary* circuits withstand a higher current density than host circuits. Currently, the prognostic cells are available for semiconductor failure mechanisms such as electrostatic discharge (ESD), hot carrier, metal migration, dielectric breakdown, and radiation effects. Other examples are presented in Section IV of this paper.

*C. Load error-seeding canaries*

Failure mechanisms initiate and grow in *canaries* when they are exposed to changes in environmental and usage loading conditions. Due to the exposure to different load level, materials or elements can deteriorate at different rates. In most cases, critical components at high-risk are assembled at a position close to the corner of a substrate. They fail to perform to set specifications in comparison to those components assembled at normal positions.

## IV. DEMONSTRATION OF EXPENDABLE *CANARIES*

Three examples of expendable *canaries* are presented in this paper: (A) Thermal aging of filaments; (B) thermomechanical fatigue of solder joints; (C) electrochemical migration (ECM) on a printed wiring board.

A. Filament *Canary:* Filaments are used in incandescent light bulbs and in X-Ray tubes to utilize electrical energy to generate photons of desired frequency range. However, some of the input electrical energy is also converted to thermal energy which results in long-term degradation and

 ISBN: 978-1-61208-109-0 

failure of the filament. In this example we explore the concept of an error-seeded *canary* that is designed to age faster than the functional filament. The test vehicle in this simple demonstration is a commercial 100mA electrical fuse. As shown in Figure 4, the *canary* is error-seeded by compromising the hermetic seal of the glass housing with a small crack which exposes the filament to oxygen in the external atmosphere.

Figure 4. *Canary* fuse with cracked glass housing.

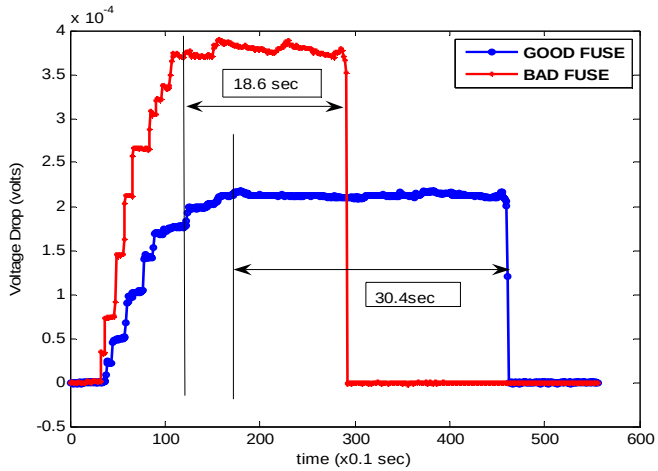



Figure 5. At 1.4 Amps: Functional filament fails at $t_1$= 30.4 secs, *canary* filament fails at $t_2$ =18.6 secs.

The flow of current causes the *canary* filament to oxidize faster than the functional filament, causing an increase in the resistivity of the material. Thus, this is an example of load error-seeding. (i.e. the *canary* filament experiences a more harsh chemical environment than the functional filament does). A constant current is supplied to both the test filament and the *canary* filament, causing increased Joule heating and faster melt-down of the *canary*. Figure 5 shows that the *canary* degrades 60% faster than the functional filament in this case, under a constant 1.4 Amps supply. In other words, when the *canary* fails, the RUL (or prognostic distance) is approximately 66% of the *canary* life.

B. Solder Joint *Canary*

Solder interconnections of electronic equipment are subjected to cyclic thermomechanical loading caused by thermal expansion mismatches during environment and operational temperature excursions. This cyclic loading can degrade solder interconnects by cyclic fatigue and eventually lead to a wearout failure. PoF models have been developed to accurately predict solder joint interconnect reliability under imposed test conditions [32-34]. However, in actual use, conditions may vary from anticipated design criteria and across field electronic equipment. In this example, two solder *canaries* are discussed: one by load-seeding and the other by geometric error-seeding [8].

The solder interconnect *canaries* presented here are specially designed to fail by the same failure mechanism and at a pre-defined prognostic distance from the failure of the functional solder joint. The proposed *canary* structures can be an extra non-functional error-seeded interconnect on a BGA device (as depicted in 6). Alternatively, it can be an additional nonfunctional component with error-seeded interconnects, such as the leadless ceramic chip resistor (LCR) depicted in Figure 7.

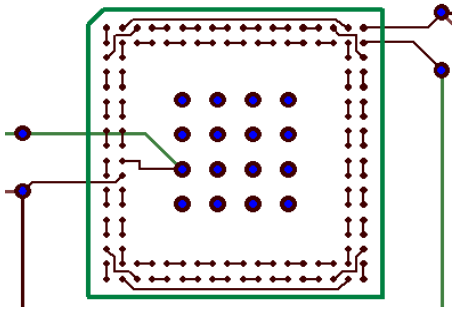


Figure 6. Example of a BGA Test Vehicle.

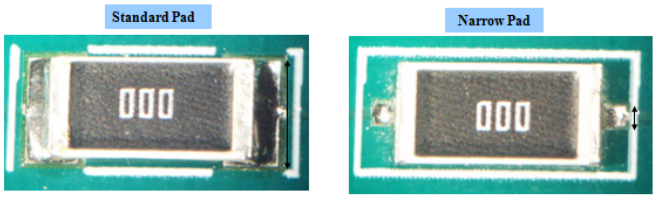



Figure 7. Resistor *Canary* Structure.

The BGA *canary* uses load seeding (due to increased distance from the component center) while the LCR *canary* uses geometric seeding by using a narrower joint pad.

Testing of BGA structure has demonstrated that in many BGA architectures, the outer solder joints may fail sooner than interior solder joints. Figure 8 shows the probability density functions of the time to failure data for the BGA depicted in Figure 6, subjected to a defined temperature cycle loading condition. Comparing the distributions reveals a less than 1% probability that an inner net will fail prior to an outer net. Comparing the mean cycles to failure reveals a prognostic distance of 1360 cycles for the given BGA test specimen and given loading condition.

Thermal cycling tests on LCRs with standard and narrow pads demonstrates a reduced life expectancy for the *canary* LCR. The proability density functions of the failure data for standard and *canary* LCRs under a defined temperature cycling test is presented in Figure 9. A comparison of distributions reveals the probability of a standard pad failure prior to a *canary* pad failure to be less than 0.01%. Examination of mean cycles to failure reveals a prognostic distance of approximately 1660 cycles.

 ISBN: 978-1-61208-109-0 

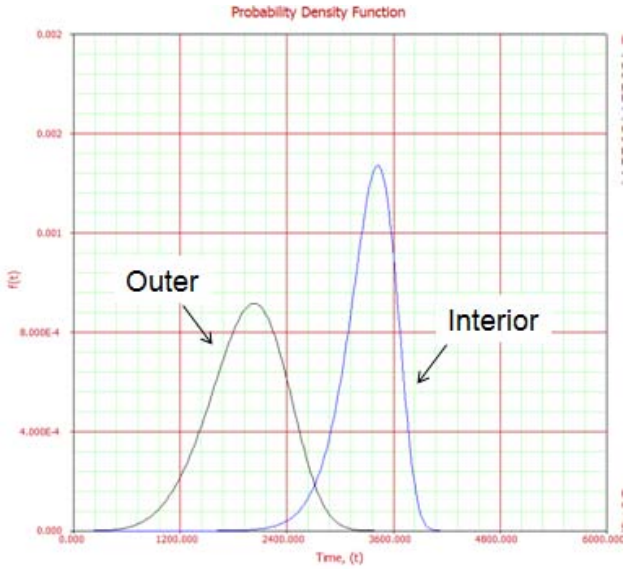



Figure 8. Probability Density Function of Cycles to Failure for an outer net and interior nets of a BGA.

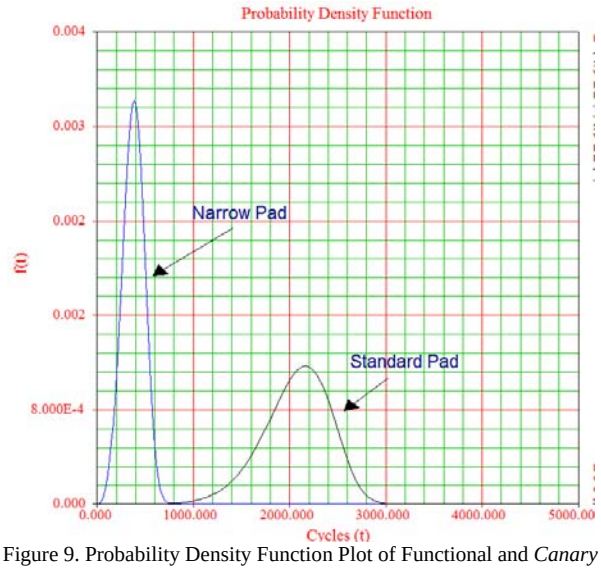



Figure 9. Probability Density Function Plot of Functional and *Canary* LCRs.

While the presented test data shows promise for solder joint *canaries*, more work is required. Testing of large IO BGAs has demonstrated that the corner solder joints are not always the first to fail. Figure 10 depicts a quarter of a 1500 I/O BGA subjected to a specific temperature cycling test. Different solder joints at different distances from the center were monitored in different daisy-chain nets shown by the different color codes. Figure 10 also depicts the frequency of first failures. As can be observed, a *canary* that only includes the three corner solder interconnects in this case would frequently miss failure prior to functional interior solder interconnects [35].

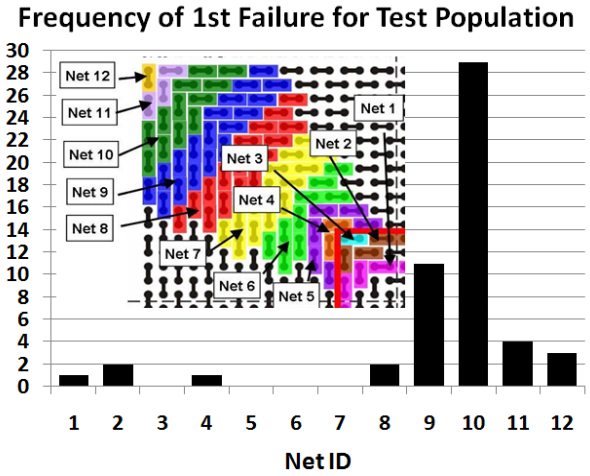



Figure 10. Frequency of First Failure in large BGA.

*C.* PWB Metallization *Canaries*:

Electronic products whose usage conditions include operation in environments with elevated levels of humidity are known to exhibit failures involving reduced surface insulation resistance (SIR). This can result in leakage currents or short circuits between portions of the circuit that are at different potential levels and are intended to be electrically isolated from one another, such as between a signal or power circuit and ground. One of the mechanisms by which the SIR of a printed circuit board (PCB) can degrade is known as electrochemical migration (ECM). The most visible manifestation of ECM is the presence of conductive metallic dendrites spanning the dielectric gap between differently biased leads, traces, or electrodes on the surface of a PCB. ECM occurs by the following sequence of steps [36]: formation of a path for ion migration between electrically biased metallic conductors, ingress into the path of a layer of electrolyte with dissolved ions typically in an adsorbed or condensed layer of water; ionization and dissolution of metal from the more positively charged conductor (anode); migration of metal cations through the electrolyte; deposition and reduction of metal cations onto the more negatively charged conductor (cathode); and growth of conductive metallic filaments towards the anode as more metal is deposited on the cathode.

*Canaries* for ECM can be incorporated into printed circuit boards and monitored for accelerated degradation in surface insulation resistance that will precede a functional failure of the circuit. The error-seeding for the *canary* design would thus need to be based on one or more known acceleration factors for this mechanism. Factors which could be used to accelerate ECM of *canary* structures include load parameters such as voltage, adsorbed moisture, and ionic contamination; geometric parameters such as spacing between conductors, and length of interface between conductors; and material parameters such as the presence of absence of a solder mask. Choice of suitable factor(s) should be based upon the achievable degree of control or repeatability, their sensitivity to specific processing or environmental risks, and their ease

 ISBN: 978-1-61208-109-0 

of integration into a product based on available space, voltage, and processing limitations.

One example of a structure which could serve as an effective ECM *canary* is a comb structure in which interdigitated conductors would be biased at similar or higher voltages than the functional circuit (Figure 11a). Acceleration of ECM failure relative to the functional circuit would be obtained due to the extended length of the interface between the conductors as well as the spacing of the conductors in the comb structure compared to the smallest spacing between biased conductors present in the circuit. This is geometric error-seeding. A source of material error-seeding is the elimination of the solder mask in the *canary* (Figure 11b), as it changes reduces the time taken for moisture to diffuse to the failure site (PWB metallization). Further acceleration could be obtained by load error-seeding, such as elevating the voltage applied across the adjacent conductors. A cross-hatched solder mask pattern across the comb could accentuate sensitivity to trapped contaminants associated with solder assembly or handling and anticipate circuit failures resulting from sub-standard processing procedures. These *canary* concepts are currently under investigation in this study and results will be presented in a future paper.

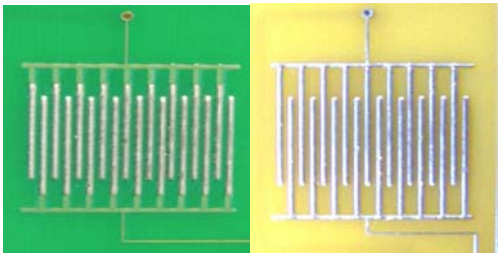



Figure 11. ECM *canaries* consisting of comb structures patterned from metallization on a PCB: (a) with solder mask between conductors; (b) with no solder mask.

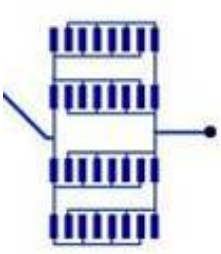


Figure 12. Conductor pattern for ECM *canary* for use with a pair of surface mount leaded packages.

Another example of a *canary* suitable for ECM is one in which the opportunity exists for dendritic growth next to or under surface mount components (Figure 12). In this case, acceleration of ECM failure can be obtained due to entrapment of moisture or ionic contaminants under the components, such as by residues of a no-clean solder flux; and/or by voltage and spacing, as well as total length of interface between the multiple adjacent solder pads connected in parallel.

## V. CONCLUSIONS

Prognostics and health management is emerging as a popular alternative to traditional reliability assessment methods, for assuring product availability. As an important approach to prognostics, *canaries* have attracted more and more attention from industry. The definition of three different types of *canaries* has been presented: expendable *canaries*, sensory *canaries* and conjugate-stress *canaries*. Three main design methods for expendable *canaries* have been described, and three examples have been presented. In the interest of space, discussion of other types of canaries is deferred to a future paper. However, there remain unanswered questions of *canaries* for PHM. For instance, PoF knowledge is needed to ensure that a *canary* will behave in the same manner as host devices and fail due to the same failure mechanisms as host devices. Since manufacturers already have mature and standard assembly lines for existing products, affordably retrofitting *canary* devices, especially with different scaling, is a big challenge. Another big challenge is the invasiveness of *canaries*, based on their impact on degradation rates of the functional host.

Research on *canary* approaches for PHM is extremely critical for electronics that must meet high availability targets. By gaining a better understanding of the design rules of *canaries*, PHM will gain more momentum and more penetration into our daily life, and ensure higher availability for commercial and military products.

## ACKNOWLEDGMENT

We thank the sponsors of the Prognostics and Health Management Consortium (PHMC) of the Center for Advanced Life Cycle Engineering (CALCE) at the University of Maryland (UMD), for supporting this study. Also, the authors would like to acknowledge Mark Zimmerman, Arvind Sai Sarathi Vasan, Lakshmi Krishnan, Vinie Somani, and Gurukrupa Kumaraswamy for their valuable help.

## REFERENCES

1. K. Keller, K. Swearingen, J. Sheahan, M. Bailey, J. Dunsdon, K.W.Przytula, and B. Jordan, “Aircraft Electrical Power Systems Prognostics and Health Management,” *IEEE Aerospace Conference Proceedings*, Big Sky, MT, March 2006.
2. A.H.C. Tsang, “Condition-based Maintenance: Tools and Decision Making”, *Journal of Quality in Maintenance Engineering*, 1(3), pp. 3-17, 1995.
3. D. K. Han, M. G. Pecht, D. K. Anand, and R. Kavetsky, “Energetic Material/Systems Prognostics,” *53rd Annual Reliability & Maintainability Symposium (RAMS)*, 2007.
4. S. Mishra and M. Pecht, “In-situ sensors for product reliability monitoring,” *Proceedings of the SPIE 2002*; 4755, pp. 10–19.
5. Ridgetop Semiconductor-Sentinel Silicon™ Library, “Hot Carrier (HC) Prognostic Cell,” 2004; August.

 ISBN: 978-1-61208-109-0 

6. J. Wang and B. Calhoun, "Canary Replica Feedback for Near-DRV Standby VDD Scaling in a 90nm SRAM," *IEEE 2007 Custom Integ Circuits Conf* (CICC), pp. 29-32.
7. B.H. Calhoun and A. Chandrakasan, "Standby Power Reduction Using Dynamic Voltage Scaling and Canary Flip-Flop Structures," *IEEE J. Solid State Circuits*, vol. 39, no. 9, Sept. 2004.
8. N. Anderson and R. Wilcoxon, "Framework for prognostics of electronic systems," *Proceedings of International Military and Aerospace/Avionics COTS Conference*, Seattle, WA, 2004; August 3–5.
9. D. Goodman, B. Vermeire, J. Ralston-Good, and R. Graves, "A Board-Level Prognostic Monitor for MOSFET TDDB," *IEEE Aerospace Conference*, March 2006.
10. M. Pecht, *Prognostics and Health Management of Electronics*, Wiley- Interscience, August 2008.
11. Telcordia Technologies, Special Report SR-332, "Reliability Prediction Procedure for Electronic Equipment," *Issue 1, Telcordia Customer Service*, Piscataway, NJ, 2001.
12. W. Denson, "A Tutorial: PRIS," *RAC Journal*, 1999, pp. 1-6.
13. FIDES Group, "*FIDES Guide issue A, Reliability Methodology for Electronic Systems*," 2004.
14. M. Cushing, D. Mortin, J. Stadterman, and A. Malhotra, "Comparison of Electronics-Reliability Assessment Approaches," *IEEE Transactions on Reliability*, Vol. 42, No. 4, 1993, pp. 542-546.
15. S. F. Morris, "Use and Application of MIL-HDBK-217," *Solid State Technology*, 1990, pp. 65-69.
16. J. Gu and M. Pecht, "Prognostics and Health Management Using Physics-of-Failure," *54th Annual Reliability & Maintainability Symposium*, Las Vegas, NV, 2008.
17. A. Strachan, E. Kober, A. van Duin, J. Oxgaard, and W. Goddard III, "Thermal Decomposition of RDX from Reactive Molecular Dynamics," *Journal of Chemical Physics*, 122, 054502, 2005.
18. M. Pecht, R. Radojcic, and G. Rao, *Guidebook for Managing Silicon Chip Reliability*, CRC Press, Boca Raton, FL, 1999.
19. P. Wang, and G. Vachtsevanos, "Fault prognostics using dynamic wavelet neural networks," *Artificial Intelligence for Engineering Design and Manufacturing*, 15, 2001, pp. 349-365.
20. T. Brotherton, J. Jahns, J. Jacobs, and D. Wroblewski, "Prognosis of Faults in Gas Turbine Engines," *Proceedings of the IEEE Aerospace Conference*, IEEE, New York, Vol. 6, 2000, pp. 163 –171.
21. B. Parker, Jr., T. Nigro, M. Carley, R. Barron, D. Ward, H. Poor, D. Rock, and T. DuBois, "Helicopter gearbox diagnostics and prognostics using vibration signature analysis," *Proc. SPIE—The International Society for Optical Engineering*, Vol. 1965, 1993, pp. 531–542.
22. C. Byington, P. Kalgren, R. Johns, and R. Beers, "Embedded diagnostic/prognostic reasoning and information continuity for improved avionics maintenance," *Proceedings of AUTOTESTCON 2003: IEEE Systems Readiness Technology Conference*, 2003, pp. 320-329.
23. W. Fenton, L. Maguire, and T. McGinnity, "Fault diagnosis of electronic systems using intelligent techniques: A review," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol 31, no 3, August, 2001, pp. 269-281.
24. N. Dev and B. Anderson, "Pimtool: An expert system to troubleshoot computer hardware failures," *Proc. Innovative Applicat. Artif. Intell.*, 1997.
25. O. Duffaut and C. Castel, "A method to aid test-tree generation for hardware fault isolation in automotive electronic control systems," in *Proc.TOOLDIAG Int. Conf. Fault Diagnosis*, 1993, pp. 230-237.
26. A. Siddique, G. Yadava, and B. Singh, "Applications of artificial intelligence techniques for induction machine stator fault diagnostics: review," *4th IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives*, 2003, pp. 29- 34, 24-26 Aug. 2003.
27. K. Przytula and D. Thompson, "Development of Bayesian diagnostic models using troubleshooting flow diagrams," *Proc. SPIE*, Vol. 4389, p. 110-120, July 2001.
28. A. AL-Jumah and T. Arslan, "Artificial neural network based multiple fault diagnosis in digital circuits," in *Proc. ICCAS*, vol. 2, pp. 304-307, 1998.
29. S. Ganesan, V. Eveloy, D. Das, and M. Pecht, "Identification and Utilization of Failure Mechanisms to Enhance FMEA and FMECA," *Proceedings, IEEE Workshop on Accelerated Stress Testing and Reliability (ASTR)*, Austin, TX, October 3-5, 2005.
30. A. Ramakrishnan, T. Syrus, and M. Pecht, "Electronics Hardware Reliability," Avionics handbook, CRC Press, Boca Raton, FL, pp.22.1-22.21, December 2000,.
31. N. Vichare, and M. Pecht, "Prognostics and Health Management of Electronics," IEEE Transactions on Components and Packaging Technologies, Vol. 29, No. 1, March 2006, pp. 222-229.
32. Q. Zhang, A. Dasgupta, P. Haswell, "Partitioned viscoplastic-constitutive properties of the Pb-free Sn3.9Ag0.6Cu solder," *J Electronic Materials* 2004; 33(11), pp,1338–49.
33. M. Osterman and M. Pecht, "Strain Range Fatigue Life Assessment of Lead-free Solder Interconnects Subject to Temperature Cycle Loading**,"** *Soldering & surface Mount Technology*, Vol. 19, No. 2, 2007, pp. 12-17.
34. A. Dasgupta, C. Oyan, M. Pecht and D. Barker, "Solder Creep-Fatigue Analysis by an Energy-Partitioning Approach**,"** *AMSE Journal of Electronic Packaging*, Vol. 144, June 1992, pp. 152-160.
35. S. London, D. Fricano, A. Dasgupta, T. Reinikainen, G. Freitas, and C. Pagliosa, "Probabilistic effects in thermal cycling failures of high-I/O BGA assemblies," *10th International Conference on Thermal, Mechanical and Multi-Physics simulation and Experiments in Microelectronics and Microsystems, Proc. IEEE EuroSimE*, April, 2009, pp. 26-29, DOI 10.1109/ESIME.2009.4938418.
36. S. Zhan, M. Azarian, and M. Pecht, "Surface Insulation Resistance of Conformally Coated Printed Circuit Boards Processed With No-Clean Flux," *IEEE Transactions on Electronics Packaging Manufacturing*, Vol. 29, Issue 3, July 2006, pp. 217 - 223.


 ISBN: 978-1-61208-109-0 

# ESPranto: a Framework for Developing Applications for Adaptive Hardware

Robert van Herk & Leszek Holenderski
*Human Interaction & Experiences*
*Philips Research*
*Eindhoven, the Netherlands*
*Email: robert.van.herk@philips.com, leszek.holenderski@philips.com*

***Abstract*—Evermore highly adaptive hardware toolkits become available, of which the applications are configured, fine-tuned or even created altogether by their end-users. To support these users, we need to think about flexible, yet accessible frameworks with which these end-users can adapt applications to their personal needs. We present a programming framework, called ESPranto, which strives exactly to do so. ESPranto is basically Esterel extended with funcros. Esterel is a reactive programming language suited for programming control-dominated applications. Commands can be executed in parallel and parallel commands can communicate. Contrary to most language, the Esterel compiler automatically proves that programs will never cause run time problems such as deadlocks, race conditions or crashes. Because Esterel does not have very strong abstraction mechanisms we added funcros. Funcros are functional macros. Like macros, they are statically expanded when applied to actual arguments. Unlike macros, funcros are type safe and hygienic (local declarations are renamed during expansion, to guarantee that there are no collisions with existing identifiers). Funcros can be recursive and higher-order in the sense that they take funcros as parameters and return funcros as results. One can also apply them partially, by providing less actual parameters than formal parameters. In addition, we added a polymorphic type system to Esterel, to allow funcros to be polymorphic and hence make ESPranto more expressive, while maintaining type safety. These extensions allowed us to use ESPranto as a host language for embedding several domain specific languages. The languages were specifically designed for developing reactive applications for a storytelling environment called StoryToy and for a tangible interaction tablet called TagTiles. The macro-like properties allowed us to keep the useful features of Esterel described above. The functional properties allowed us to use ESPranto to facilitate end-user programming: our end-users use ESPranto to adapt and extend applications to their own needs.**

***Keywords*-adaptive systems; computer languages; parallel programming; functional programming.**

## I. INTRODUCTION

Recently the world has seen a huge rise in commercially available, highly adaptive hardware toolkits, such as the iCat [1], Arduino [2], Phidgets [3] and Lego MindStorms [4]. Whereas in classical computing a rather fixed number of applications suffices for most users, for these toolkits the applications are configured, adapted or even created altogether by end-users. These application creators may be non-technical domain experts – for instance experts in human robot interaction that use these products for their research [1] – or laymen, such as technical hobbyists, children, teachers or parents [2].

Figure 1. StoryToy

Enabling all these users to easily adapt such hardware to their own needs, whist giving them access to all functionality, is a daunting task, but some attempts are being made [1]–[3]. However, we still face the issue that adaptation frameworks that are powerful and truly accessible to non-technical users hardly exist for such hardware toolkits.

This paper describes the development and usage of the programming language ESPranto. We apply ESPranto especially to embed domain specific languages (DSLs) used by end-users to fine-tune or create applications for adaptive hardware toolkits. We called the language ESPranto because it is a language for "Edutainment Sensor Platforms (ESPs)" and because it strives to be applicable to a diverse class of users.

One toolkit for which the language is used is StoryToy [5] (Fig. 1). A second system is TagTiles [6] (Fig. 2), marketed by Serious Toys [7].

StoryToy is a toolkit for creating storytelling applications. These applications can entertain and promote the development of young children (2-6). The child implicitly provides input to the system by shaking fluffy farm animals, thus

 ISBN: 978-1-61208-109-0 

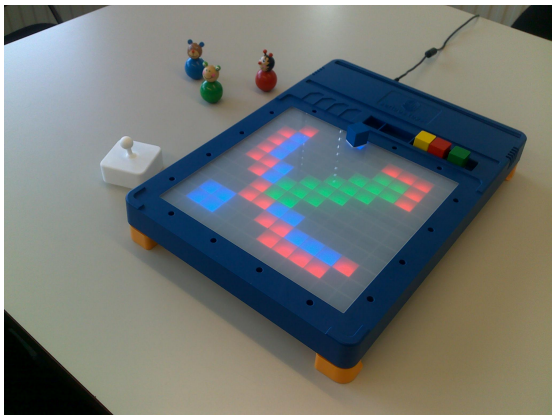


Figure 2. TagTiles

guiding the plot of the story. Lighting support is available for drama. For example, one created application was set in a thunderstorm, featured lightning effects. We have found that laymen users, such as parents or teachers, enjoy adapting StoryToy to specific children by making personal applications (stories) for them [8].

TagTiles is a tangible interface console designed for educational board games (Fig. 2). It features an antenna grid that identifies and localizes objects tagged with RFID tags on the two-dimensional surface. For providing feedback, every grid cell is equipped with a full color LED and the board can produce audio like speech, sound effects or music. Also, the console is equipped with an embedded computer. Non-technical domain experts, e.g., psychologists, are using TagTiles in their research [6], [9]. Also TagTiles applications can be adapted by teachers for usage in their classrooms [8].

Before creating the ESPranto language, we conducted a series of studies in which domain experts and laymen were asked to design a range of applications for adaptive hardware such as StoryToy and TagTiles [8]. We found that participants tend to describe these applications as 'story lines' that develop in parallel. Each story line is described sequentially and in some cases must be preempted (i.e., aborted). Furthermore, the application domains the different users came up with were very diverse [6], [9], [10]. Domain experts and laymen indicated that they want to spend as little time as possible addressing technical issues. From these sessions, we concluded that ESPranto needed the following features:

1) To keep the language close to the users' mental model, parallelism, sequencing and preemption must be supported.
2) ESPranto must enable embedding of a multitude of domains.
3) Errors must be given in terms of the end-user's domain.
4) Run time problems such as crashes, non deterministic behavior, or deadlocks must be prevented.

Implementing ESPranto as an Esterel-like language seemed an obvious choice: Esterel supports parallelism and preemption [11], meeting requirement 1. Also, Esterel applications will never cause run time errors, meeting requirement 4. However, we argue that Esterel *per se* is not suitable for implementing DSLs because it lacks crucial abstraction mechanisms, needed for requirement 2 and 3. Many DSLs have been successfully implemented in modern functional programming languages that do provide those crucial abstraction mechanisms.

To meet all requirements, we designed ESPranto as a subset of Esterel, extended with the required features from functional programming. ESPranto is based on the concept of 'funcros'. Funcros are like functions in functional programming and at the same time like macros. Because of the power of funcros, ESPranto maintains the core benefits of Esterel but introduces the benefits of functional programming, thus making it suitable for embedding DSLs for tangible applications. We have embedded a series of DSLs in ESPranto and these DSLs are successfully used by domain experts to create tangible applications.

The remainder of this paper is structured as follows. In Section II, we go deeper into DSLs and explain how they are embedded into functional programming languages. Section III gives a basic introduction into reactive programming in Esterel. Section IV describes previous work on combining reactive programming with functional programming. In Section V, we explain how ESPranto can be translated into Esterel, yet unleashes the power of functional programming through funcros. In Section VI, we provide examples of DSLs that are embedded in ESPranto and show how ESPranto's features are applied by non-technical end-users to create and adapt applications. We will use examples of TagTiles applications, because they are most advanced and are indicative of ESPranto's strength. In Section VII, we present our conclusions.

## II. Domain Specific Languages

Mainstream programming languages (like C and Java) do not enable domain experts who are not professional programmers to adapt applications. Therefore, DSLs have been designed, which enable domain experts to develop applications for their particular domain quickly and effectively, yielding programs that are easy to understand, reason about, and maintain by the domain experts themselves [12].

Designing a programming language from scratch is difficult and time-consuming. Effort is wasted, since large parts of many DSLs are not domain specific at all. For instance, many require a type system and a sub-language for integer and Boolean expressions. For these reasons, language designers often *embed* a DSL as a library into an existing *host* language. Because a Domain Specific Embedded Language (DSEL) inherits features from its host language, the chosen

 ISBN: 978-1-61208-109-0 

host language must be flexible enough to allow for a natural embedding of the DSL, and its features must be suitable for, or at least not at odds with, the properties of the domain.

Modern functional programming languages are often a sensible choice when crafting DSELs for creating desktop computing applications; examples are abundant [12]. Functional languages provide good abstraction mechanisms due to polymorphic type systems and, as an alternative to statements, the flexible concept of functions. For example, polymorphic functions allow to define generic statements. Higher order functions allow to pass statements as parameters, allowing for yet stronger abstractions. Partial function application allows to use specialization, by defining a function in terms of another function with only some of its parameters provided. For instance, a function `inc` can be defined as `(+) 1`.

## III. ESTEREL

Although it is generally easy to embed DSLs in functional programming languages, functional programming *per se* is not suitable for embedding DSLs for *tangible* computing: functional programs cannot be checked for causal correctness, are not real-time and not optimal for resource constrained systems. Reactive programming languages provide these features, but lack flexibility. Before describing how we combined both paradigms, we will first give a basic introduction into Esterel by discussing some examples. For a full explanation of Esterel we refer to [11].

Esterel is based on the two basic concepts of *commands* and *signals*. Commands describe control-flow and signals are used for communication with hardware or between commands.

An external control loop is in place that decides when the Esterel program is allowed to execute; usually when the underlying hardware has detected some physical event or when a predetermined amount of time has passed. Then, the Esterel program computes a reaction, typically in a very short amount of time, and pauses again. One such an iteration is called an *instant*. When the control loop decides to run the next instant, the Esterel program will resume execution at the point at which it stopped in the previous instant.

In every instant, for every signal it holds that that signal is either present or absent. With this feature Esterel provides *causal correctness*, which means that race conditions cannot occur because all communication within an instant appears to be instantaneous.

Esterel is crafted such that an instant always runs in a finite amount of time. Because of this feature, Esterel is a *real-time* programming language. This requires that all commands that *wait* for some event to happen end the current instant and pass control back to the control loop.

After giving a few introductory examples, we will briefly discuss the implications of causal correctness and real-timeness.

```
//Declaration of signals
input ButtonPress
output SoundBell

//The control code itself:
await ButtonPress;
emit SoundBell
```

Figure 3. Esterel program controlling a doorbell

### A. Controlling a simple machine

A trivial example of an Esterel program is given in Figure 3. This program controls a simple doorbell. The program will make the bell sound once when the button is pressed and then terminate.

The program has two signals. `ButtonPress` is an *input* signal that describes the state of a sensor: when the button is pressed the control loop sets the signal to present and then makes the Esterel program run for one instant. `SoundBell` is an *output* signal: after the Esterel program has run for one instant, the control loop decides to sound the bell or not depending on the presence of this signal.

When the external control loop makes this program react for the first time, it will immediately end the instant because control reaches the `await` command. Hence, in the first instant the output signal `SoundBell` is absent, meaning that the external control loop should not sound the bell.

When the control loop makes the program react for the second time, it will resume at the point where it stopped in the first instant; i.e., at the `await` command. Two things can happen, depending on the presence of the input signal `ButtonPress`. If `ButtonPress` is absent, the program will immediately end the instant again, as it is waiting for an instant in which `ButtonPress` is present. When an instant is run in which `ButtonPress` is present, the `await` command terminates and the program continues and emits `SoundBell`. Since that is the last command of the program, it will then terminate. The control loop reacts to the presence of `SoundBell` by ringing the bell.

### B. Parallelism and communication

The example given in Figure 4 controls a more advanced doorbell and uses parallelism and communication. In this example, the doorbell has two buttons and has two different melodies. The program turn-wise plays one of these melodies when button 1 and/or button 2 is pressed, but it will never interrupt a playing melody.

We have introduced an input signal `MelodyDone` that the control loop will set to be present in every instant in which a melody has just stopped playing. The program has three loops. The three loops will run in parallel because

 ISBN: 978-1-61208-109-0 

```
input Button1Press
input Button2Press
input MelodyDone
output PlayMelody1
output PlayMelody2

signal GenerateOuput in
  loop
    await Button1Press;
    emit GenerateOutput
  end loop
  |
  loop
    await Button2Press;
    emit GenerateOutput
  end loop
  |
  loop
    await GenerateOutput;
    emit PlayMelody1;
    await MelodyDone;
    await GenerateOutput;
    emit PlayMelody2;
    await MelodyDone
  end loop
end signal
```

Figure 4. A more advanced doorbell

they are combined with the parallel operator `|`. The loops communicate through the *local* signal `GenerateOutput`.

The upper loop waits for an instant in which `Button1Press` is present. In such an instant, it will emit `GenerateOutput` and then restart the loop again. Restarting the loop will cause the upper loop to end for this instant, because of the `await` command. The second loop reacts likewise on the presence of `Button2Press`.

The third loop waits for an instant in which the local signal `GenerateOutput` is present. In the first instant in which that happens, it will react by emitting `PlayMelody1`. The program will then wait until the melody has stopped playing. This prevents the program from ever starting a new melody when a melody is already playing. Only after the melody has stopped, it will wait for an instant in which `GenerateOutput` is present again and then react by emitting `PlayMelody2`. When that melody has finished, the third loop will be restarted and wait again for `GenerateOutput`.

### C. Implications of causal correctness and real-timeness

In Esterel writing such control-dominated programs is easy: separate parts of the program can be defined separately and composed in parallel, parallel commands can communicate through signals but race conditions will not occur, no variables are needed to keep track of the state of the program and the program will never deadlock because all instants will always end after a finite amount of time.

In imperative languages such as C, or functional languages such as Haskell, writing a similar program correctly is more difficult.

One approach would be to use asynchronous multi-threading and shared memory. This would require the programmer to *synchronize* when accessing the shared memory to prevent race-conditions, and to prove that deadlocks will not occur due to this synchronization.

Another approach would be to use event-handling procedures that are called when the buttons are pressed or when a melody has finished playing. In that case, one would need to introduce variables to keep track of which melody is currently playing, if any. The programmer would need to think about all possible interleavings of events by himself, which can be difficult. Furthermore, the programmer would have to construct a proof that the event-handling procedures always terminate such that new events can continue to be accepted from the hardware.

```
signal S in
  if S
    then nothing
    else emit S
  end if
end signal
```

Figure 5. Causally incorrect Esterel program

### D. Incorrect programs

The Esterel compiler rejects programs of which causal correctness cannot be proved. An example of such an erroneous program is given in Figure 5. The program is erroneous, because we cannot decide on the presence of `S`: if `S` is absent in the instant in which this program is started, `S` has to be emitted which means that it is present in that instant. Conversely, when `S` is present in the instant in which this program is started, it is never emitted, which means that it is absent in that instant.

Another example of an incorrect Esterel program is given in Figure 6. This program is causally correct, but still erroneous because the body of the loop does not contain any command that ends the instant. This means that we cannot establish a finite run time for the instant in which the loop starts, and therefore the Esterel compiler will reject this program. A correct version of this program pauses after emitting `S`. The command `pause` ends the current instant and terminates immediately in the next instant.

 ISBN: 978-1-61208-109-0 

```
//Erroneous program
signal S in
  loop
    emit S
  end loop
end signal

-----------------------------------------

//Correct program
signal S in
  loop
    emit S;
    pause
  end loop
end signal
```

Figure 6. Esterel program with instantaneous loop

### E. Limitations

Although Esterel provides very strong correctness properties, it provides few abstraction mechanisms. It is not possible to define functions or procedures. If such mechanisms were in place, the Esterel compiler could not automatically check for causal correctness, nor check that every instant has finite run times, because this is undecidable for Turing complete languages [13].

Some modularity can be achieved through a mechanism called *modules*, which are textually expanded. Modules can only be parametrized with signals and not with commands or other modules, nor can they be applied recursively or partially.

## IV. Combining functional and reactive programming

As functional programming provides flexibility, and reactive programming provides causal correctness and real-timeness, ESPranto strives to combine best of both paradigms such that it can be used to embed DSLs for tangible computing.

A combination of functional programming with reactive programming was first proposed in Lucid Synchrone [14], a language that extends OCaml with some concepts borrowed from a data-flow synchronous language Lustre [15]. There are two main differences between Lucid Synchrone and ESPranto.

The first main difference is that our combination uses a different order in which the one programming paradigm extends the other. Unlike Lucid Synchrone, we extend a synchronous language with some concepts borrowed from functional languages. This difference has two important consequences. First, for our class of users, the reactive programming paradigm is much more natural than the functional one (requirement 1). Therefore ESPranto allows novices to start with reactive programming, and later on learn to fully use the additional functional features funcros provide for their convenience. This is of the utmost importance when dealing with application developers who are not professional programmers. Second, we can reduce substantially the runtime resources required to execute ESPranto programs, compared with Lucid Synchrone. We first compile ESPranto programs to Esterel, by expanding funcro applications, and then compile Esterel to assembly code. In the approach taken by Lucid Synchrone, programs are first expanded to OCaml, and then compiled by the OCaml compiler. This approach requires a much larger execution footprint, especially in terms of memory due to the need for garbage collection, which is inherent to functional programs.

```
/* ESPranto program */
main = (switch true)
switch c = (if c then pause else halt)

-----------------------------------------

/* Esterel translation */
if true then pause else halt
```

Figure 7. A trivial ESPranto program and its Esterel translation

The second main difference is that ESPranto incorporates an imperative synchronous language while Lucid Synchrone incorporates a declarative one. Again, this is easier to understand for our users, as sequential composition must be a basic combinator in our language (requirement 1). The data-flow paradigm, on which Lustre is based, is not well suited to describe imperative control flow.

## V. ESPranto

### A. Funcros

Every funcro is a tuple of a name, a series of zero or more formal parameters and a body. Funcros can be applied by providing them with actual parameters. The ESPranto compiler expands a funcro with all its actual arguments into basic Esterel statements. A trivial example of how this works is shown in Figure 7. Expansion always starts at the funcro called `main`. For the reader unfamiliar with Esterel, an overview of the most important basic statements is given in Appendix A.

When expanding a funcro application, formal parameters are substituted for their values. From this perspective, funcros are like macros as available in languages like C. ESPranto uses expansion (like macros) instead of dynamic application (like functions) to substantially simplify the causal correctness problem. If we used functions, but still wanted to keep Esterel's strong correctness properties, we would be faced with the problem of compositional verification of causal correctness. We are not aware of any method that solves that problem.

 ISBN: 978-1-61208-109-0 

```
#define INC(i) {int b = 17; ++i;}
void weird () {
    int a = 0; int b = 0;
    INC(a); // a becomes 1.
    INC(b); // b remains 0.
}
```

Figure 8. Unhygienic macro expansion in C

Unlike C macros, funcros are type safe, hygienic, and ESPranto supports recursion, higher order funcros, and partial funcro application.

### B. Type safety

The ESPranto compiler automatically infers the type of each funcro, using the Hindley-Milner type inference algorithm [16]. Before expanding a program into Esterel code, it checks if all funcro applications are type correct. If the program is type incorrect, the compiler will provide feedback on this to the user. Because types are checked *before* funcro expansion, the compiler gives feedback in terms of the domain specific funcros, and not in terms of the Esterel code. This is needed to meet requirement 3. Only if it is type correct, the compiler will produce the corresponding Esterel program that can be compiled into machine code in the next step. The `main` funcro must always have type `Command`, i.e., it must expand to a valid Esterel command.

For the code segment of Fig. 7, the compiler infers that `switch :: Condition → Command` (`::` should be read as *has type*, in other words `switch` takes a `Condition` as a parameter and then unfolds into a `Command`) because `c` is used as a condition in its body and because its body unfolds into an if-statement. Likewise the compiler infers that `main :: Command`, making the example program type correct.

### C. Hygienic expansion

A known problem with 'naive' macro expansion systems is that identifiers may collide. Languages featuring such macro expansion systems are called 'unhygienic' [17]. An example is the C programming language (see Fig. 8). Programmers typically try to solve hygiene problems by *obfuscation*: use only unusual variable names in macros and hope that the same names will never be used in the rest of the program.

Obfuscation is not a suitable solution for ESPranto. It is a delicate and error prone method that requires the domain expert to inspect all used funcros. Instead, ESPranto has two features that together prevent hygiene problems. The first feature is scope: identifiers such as signals or traps are *scoped* within the funcro in which they are declared. They can be used outside their declaring funcros only by passing them explicitly as parameters to other funcros.

```
/* ESPranto code */
main = (
  trap T (
      myFuncro T //Explicitly pass T
  )
)
myFuncro TrapLabel = (
  trap T (
    exit TrapLabel
  )
)

-------------------------------------

/* Esterel translation */
trap T_0 in
  trap T_1 in
     exit T_0
  end trap
end trap
```

Figure 9. Example showing ESPranto's hygienic transformation

The second feature is the manner of transformation into Esterel code, in which Esterel identifier clashes are prevented by augmenting signal and trap labels with their *call depth*. The call depth is defined as the number of funcro applications in which the currently expanded funcro application is nested. The call depth of `main` is 0, and subsequently increases when calls are nested. Examples of how this works are given in Fig. 9. Note how the trap label in the exit statement is augmented with call depth 0 in the Esterel transformation, correctly referring to the outer trap.

### D. Recursive funcro definitions

Most programming languages do not support recursively defined macros, because naively substituting macro applications with their expanded bodies would lead to an infinite amount of expanded code for recursive definitions. However, recursive functions is one of the essential features of functional programming that we wanted to include in ESPranto. We resolved this issue by allowing programmers to define alternative expansions of a funcro, guarded by *patterns*. For each funcro application, the compiler matches actual parameters against the patterns of the alternatives. For instance, the pattern $i$, where $i$ is an identifier, matches against any parameter, the pattern `{}` matches only against an empty list, and the pattern $p_1 : p_2$, where $p_1$ and $p_2$ are patterns, matches only against a non-empty list, if the first element of that list matches against $p_1$ and the remainder of the list against $p_2$.

It is still possible to define recursive funcros that generate an infinite amount of code when they are expanded. An example is given in Fig. 10. We are currently investigating

 ISBN: 978-1-61208-109-0 

```
/* ESPranto code */
length (h:t) =
  (1 + length (h:t)) // Wrong!
length {}    =
  (0)   // Non-recursive alternative

//This expansion will not terminate:
main = (
  if length {1,2,3} = 3
    then nothing
    else nothing
)
```

---

```
/* ESPranto code */
length (h:t) =
  (1 + length t) // Right
length {}    =
  (0)    // Non-recursive alternative

//This expansion will terminate:
main = (
  if length {1,2,3} = 3
    then nothing
    else nothing
)
```

Figure 10. Example showing recursive funcros

```
<funcrodef>  ::=  <identifier> <patterns>
                  = (<expr>)
  <pattern>  ::=  <identifier>
               |  <pattern> : <pattern>
               |  {}
               |  etc
     <expr>  ::=  <esterel>
               |  <identifier>
               |  <funcroapp>
               |  etc
  <esterel>  ::=  nothing
               |  pause
               |  true
               |  false
               |  <number>
               |  <expr> ; <expr>
               |  <expr> | <expr>
               |  <expr> && <expr>
               |  <expr> + <expr>
               |  etc
<funcroapp>  ::=  <identifier> <exprs>
```

Figure 11. Simplified syntax of ESPranto

if we should restrict the class of allowed recursive funcros. A possible restricted class can be those funcros of which the compiler can prove that all their possible applications will generate a finite amount of code. A first idea is to allow catamorphic [18] recursive funcros only, as applying such funcros always terminates. Catamorphic funcros break down a data structure in their recursion, and have a non-recursive alternative for $\perp$. With such a restriction, the compiler would indeed reject the first program in Fig. 10, and allow the second. Similar restrictions have been successfully applied before, for instance in the Agda programming language [19].

### E. Syntax of ESPranto

For clarity, we present an excerpt of the grammar of ESPranto in Fig. 11. For clarity, the excerpt is heavily simplified. Important to note is that in ESPranto, an expression can be an Esterel literal, a funcro application, or an identifier bound through a pattern of a formal parameter. A complete description of the Esterel syntax can be found in [11].

## VI. Embedding Domain Specific Languages in ESPranto

We have successfully embedded multiple DSLs in ESPranto, each for intrinsically different domains, including physio-therapy, [9] and games that use concepts from geometry [10].

To demonstrate how ESPranto's features facilitate end-users to adapt and extend applications, we will go deeper into one DSEL. This DSEL is created for a set of related exercises for the TagTiles console, that are, somewhat confusingly, called TagTiles Classic [6].

### A. The domain of TagTiles Classic

The goal of TagTiles Classic was to create fun, educational exercises for children, which address cognitive and motor skills. In the vocabulary of the domain expert (psychologist), TagTiles Classic consists of a series of tasks and each task consists of a series of levels. All levels in TagTiles Classic start with displaying an *assignment pattern* by lighting up cells in particular colors on the TagTiles board. They then require the child to indicate a *solution pattern* by *tagging* cells with colored game pieces. The tasks differ in how the child has to work out the solution pattern from the assignment pattern. A particular set of tasks targets training spatial insight. For instance, in the *translation task* the child has to translate assignment patterns

 ISBN: 978-1-61208-109-0 

```
//Global signals
signal LevelFinished
signal RestartLevel
level body = (
  abort SkipLevel (
    trap LevelFinished (
      loop (
        abort RestartLevel (
          body;
          exit LevelFinished
  ) ) ) )
  ;
  putBackThePieces
)
putBackThePieces = (
  ... /*some code*/
)
```

Figure 12. Example of parametrization in the TagTiles DSEL

```
drawTiles {} = (nothing)
drawTiles (coord:cs) = (
  drawTile coord;
  drawTiles cs //catamorphic recursion
)
awaitAndDrawTiles {} = (nothing)
awaitAndDrawTiles (coord:cs) = (
  ( awaitTile coord;
    drawTile coord
  )
  | awaitAndDrawTiles cs // catamorphic
)
memoryLevel coords = (
  level (
    drawTiles coords;
    waitTime 2000;
    clearBoard;
    emitSound "pling.wav";
    awaitAndDrawTiles coords
) )
```

Figure 13. Using recursive definitions for the TagTiles DSEL

4 cells horizontally. In the *mirror task*, the child has to create the symmetric counterpart of assignment patterns.

A programmer has embedded a DSL for these exercises into ESPranto and created a basic set of exemplary exercises. The psychologist then adapted this to her own needs, by fine-tuning the way the different tasks work, by adding extra tasks and by working out many levels for each task.

### B. Capturing the essence of levels with parametrization

The domain expert wanted all levels to have some commonalities, specifically:

1) when playing a certain level, it must be possible to skip that level;
2) when playing a certain level, it must be possible to restart that level;
3) at the end of each level, the child must put the pieces back into a starting position.

The programmer captured these commonalities in the DSEL by introducing a funcro `level` (Fig. 12), which is parametrized with a command `body` that contains the control code for a specific level. The compiler infers that `level :: Command → Command`: if the domain expert applies this funcro and passes it a command, `level` will augment it with the common functionality of levels.

### C. Using recursion in TagTiles

In the memory task every level first displays the assignment pattern for two seconds. Then the board is cleared and the child has to reconstruct the pattern from memory by tagging the correct cells. The child can reconstruct the pattern in any order. As he reconstructs the pattern, the cells light up again. Both the code for displaying the assignment pattern and for lighting up the solution pattern as the child reconstructs it can be neatly defined in ESPranto through *recursion* (see Fig. 13). Both funcros are defined recursively over a list. The funcro `drawTiles` takes a (static) list and creates a sequence of commands that each draw one cell. The funcro `awaitAndDrawTiles` takes a (static) list and waits for the child to put the right game piece on each coordinate and then illuminates that coordinate. Note how parallelism is used in `awaitAndDrawTiles` to allow the child to tag the cells in the solution pattern in any order.

### D. Higher order funcros and partial application

The levels of the tasks addressing spatial insight share further commonalities. E.g., both in the mirror task and in the translation task, in every level the assignment pattern is displayed for two seconds, then the TagTiles board is cleared and the child has to recreate the solution pattern by applying the correct mathematical transformation; e.g., mirroring or translation. Figure 14 shows how we managed to capture these commonalities in the `spatialInsightLevel` funcro, that takes the mathematical transformation as its parameter `f`, and takes a list of coordinates `coords`. Hence,

```
spatialInsightLevel :: (Coord→ Coord) →
                       {Coord} →
                       Command
```

where `{Coord}` should be read as: list of which the elements have type `Coord`.

Since `spatialInsightLevel` takes a funcro as its first parameter, it is of *higher order*. Likewise, `map` is a higher order funcro. The funcros `mirrorLevel` and

```
/* Defined in a library */
map f {} = {}
map f (e:l) = {f e : map f l}

/* Defined for the DSEL of TagTiles */
spatialInsightLevel f coords = (
  level (
    drawTiles coords;
    waitTime 2000;
    emitSound "pling.wav";
    awaitAndDrawTiles (map f coords)
) )

mirror coord = (...)
  /* takes a coord and mirrors it
     point-symmetrically  w.r.t
     the center of the TagTiles board */
translate coord dx dy = (...)

/* Defined by the domain expert */
mirrorLevel = (
  spatialInsightLevel mirror
)
translationLevel = (
  spatialInsightLevel (translate 4 0)
)
mirrorLevel1 = (
  mirrorLevel coords1
)
coords1 = (...)
          //Some list of coordinates
```

Figure 14. Using higher order funcros and partial application for the TagTiles DSEL

`translationLevel` are partially applied funcros: they feed a first parameter to `spatialInsightLevel` and hence both have type `{Coord}` → `Command`. The funcro `mirrorLevel1` is exemplary for how the domain expert finally defines a level by feeding a list of actual coordinates to `mirrorLevel`, resulting in a command.

## VII. Conclusions

We created ESPranto to alleviate the problems domain experts face when adapting and creating applications for adaptable hardware. ESPranto has proven to be a valuable framework to host DSLs for various adaptable hardware toolkits, for instance StoryToy and TagTiles, which is a tangible interface for educational board games. ESPranto enables the creation of a new domain specific language (DSL) when needed (for example, for a new type of game), with relatively low effort.

We have embedded several DSLs in ESPranto. Domain experts and laymen with no prior programming experience use these DSLs successfully to adapt and extend applications to their own needs.

We designed and implemented ESPranto as a programming language that combines reactive and functional programming by extending the reactive programming language Esterel with funcros and polymorphic types. Funcros are statically expanded similarly to macros. However, like functions, funcros are strongly typed, can be recursive, higher-order, and applied partially.

During compilation, an ESPranto program is expanded to an Esterel program which can then be checked for causal correctness using standard Esterel techniques. Using funcros (expanded during compile time) instead of functions (called at run time) allows us to verify causal correctness.

As such, ESPranto combines the prevention of runtime problems such as deadlocks or race-conditions that reactive programming offers, with the strong abstraction features that functional programming offers, making it very suitable for novice programming.

## Appendix

| Statement | Meaning |
|---|---|
| `nothing` | Terminates immediately |
| `pause` | Terminates in the next instant |
| `emit` $S$ | Signal $S$ is present in this instant. Terminates immediately. |
| `loop` $p$ | Starts $p$. In the instant in which $p$ terminates $p$ is started again. |
| `halt` | Equal to `loop pause` |
| `await` $c$ | Terminates in the next instant in which $c$ holds. |
| `if` $c$ `then` $p$ `else` $q$ | Starts $p$ if $c$ holds, $q$ otherwise. |
| `trap` $T$ $p$ | Starts $p$. Terminates in the instant in which $p$ terminates, or in the first instant in which $p$ executes `exit` $T$. |
| `abort` $c$ $p$ | Starts executing $p$. Terminates in the instant in which $p$ terminates, or in the next instant in which $c$ holds. |
| $p$`;` $q$ | Sequential composition. Starts executing $p$. In the instant in which $p$ terminates, starts executing $q$. Terminates in the instant in which $q$ terminates. |
| $p$ `\|` $q$ | Parallel composition. Starts executing $p$ and $q$. Terminates when both $p$ and $q$ have terminated. |

## Acknowledgments

The authors would like to thank Panos Markopoulos, Willem Fontijn, Janneke Verhaegh, Alexander Sinitsyn and Andrew Tokmakoff for their valuable input for this paper.

 ISBN: 978-1-61208-109-0 

## REFERENCES


[1] M. Saerbeck, "Software architecture for social robots," Ph.D. dissertation, Technical University of Eindhoven, 2010.

[2] E. Rosenbaum, E. Eastmond, and D. Mellis, "Empowering programmability for tangibles," in *TEI '10: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*. New York, NY, USA: ACM, 2010, pp. 357–360.

[3] S. Greenberg and C. Fitchett, "Phidgets: easy development of physical interfaces through physical widgets," in *User Interface Software and Technology*, 2001, pp. 209–218.

[4] D. Benedettelli, *Creating Cool MINDSTORMS NXT Robots (Technology in Action)*. APress, 2008.

[5] W. Fontijn and P. Mendels, "Storytoy the interactive storytelling toy," in *PerGames workshop, International Conference on Pervasive Computing*, 2005.

[6] J. Verhaegh, W. Fontijn, and H. Hoonhout, "Tagtiles: optimal challenge in educational electronics," in *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*. New York, NY, USA: ACM, 2007, pp. 187–190.

[7] "Serious Toys," http://www.serioustoys.com (last access: August 2010).

[8] R. Van Herk, J. Verhaegh, and W. F. Fontijn, "ESPranto SDK: an adaptive programming environment for tangible applications," in *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*. New York, NY, USA: ACM, 2009, pp. 849–858.

[9] Y. Li, W. Fontijn, and P. Markopoulos, "A tangible tabletop game supporting therapy of children with cerebral palsy," in *Proceedings of the 2nd International Conference on Fun and Games*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 182–193.

[10] K. Hendrix, R. Van Herk, J. Verhaegh, and P. Markopoulos, "Increasing children's social competence through games, an exploratory study," in *IDC '09: Proceedings of the 8th International Conference on Interaction Design and Children*. New York, NY, USA: ACM, 2009, pp. 182–185.

[11] G. Berry, "The Esterel v5 Language Primer," http://www-sop.inria.fr/meije/esterel/esterel-eng.html (last access: August 2010), 1999.

[12] P. Hudak, "Building domain-specific embedded languages," *ACM COMPUTING SURVEYS*, vol. 28, 1996.

[13] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society*, vol. 2, no. 42, pp. 230–265, 1936.

[14] P. Caspi and M. Pouzet, "Lucid Synchrone, a functional extension of Lustre," Université Pierre et Marie Curie, Laboratoire LIP6, Tech. Rep., 2000.

[15] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud, "The synchronous data flow programming language lustre," in *Proceedings of the IEEE*, vol. 79, Issue 9, 1991, pp. 1305–1320.

[16] B. Pierce, *Types and programming languages*. Cambridge, MA, USA: MIT Press, 2002, ch. 22.

[17] E. Kohlbecker, D. P. Friedman, M. Felleisen, and B. Duba, "Hygienic macro expansion," in *LFP '86: Proceedings of the 1986 ACM conference on LISP and functional programming*. New York, NY, USA: ACM, 1986, pp. 151–161.

[18] E. Meijer, M. Fokkinga, and R. Paterson, "Functional programming with bananas, lenses, envelopes and barbed wire," in *Proceedings of the 5th ACM conference on Functional programming languages and computer architecture*. New York, NY, USA: Springer-Verlag New York, Inc., 1991, pp. 124–144.

[19] "The Agda Wiki," http://wiki.portal.chalmers.se/agda (last access: August 2010).


 ISBN: 978-1-61208-109-0 

# Kind Parsing: An Adaptive Parsing Technique

Michal Žemlička and Pavel Šašek
*Department of Software Engineering*
*Charles University, Faculty of Mathematics and Physics*
*Prague, Czech Republic*
*Email: zemlicka@ksi.mff.cuni.cz, pavel.sasek@email.cz*

***Abstract*—Applications have many opportunities to be adaptive. One of them is the ability to control the input language(s). To achieve this objective we present an efficient semi-top-down parsing technique – kind parsing. It supports on-line (parse-time) as well as off-line extensions, restrictions or other adaptation of the accepted language. Kind parsers can be adapted quickly and the changes of the parser's structure tend to be only local. We suppose that such a technique can be very useful for adaptive system development.**

***Keywords*-extensible parsing, adaptive parsing, parse-time extension.**

## I. Introduction

Development and maintenance of adaptive applications could become simpler if adaptive techniques were used. Do we have enough usable adaptive techniques? We suppose that it would come to useful if there were more techniques available. Therefore, this paper describes one of them.

In the case that our input data are in an XML-based format, we can use standard XML techniques. As XML is extensible from the beginning, the XML-parsers and other XML-tools should be able to cope with it somehow.

But what happens if the input is not XML-based? Hand-written parsers as well as parsers generated by usual parser generators are not (reasonably easily) extensible. They are built to handle just one language – the one specified in the beginning. So, should we give up adaptivity for non-XML inputs? No!

We should use parsers that are extensible or even modifiable. The parser extensibility or modifiability can be off-line (applicable only on parsers that do not parse at the moment) or on-line (applicable also during parsing). To support adaptability or self-adaptability it could be better if the parser could be extended or modified on-line.

There is a parsing technique called *kind parsing* [1], [2] allowing on-line changes of the parsers and being efficient (parsing in the linear time). We will shortly describe the parsing technique here and show that it is possible to use its structure also for on-line language restrictions. If we have both on-line extensibility and restrictability of the input language, we get an on-line modifiable (adaptable) parsing technique.

### A. Paper Structure

It is reasonable to start projects with the specification of requirements. We collect requirements on parsers to be adaptive in Section II. The structure of Kind Parsers is introduced in Section III. Section IV presents how they work. Section V describes how Kind Parsers can be extended and when it is safe to do that. A discussion about communication of parsers with other parts is in Section VI. Issues related to an (on-line) parser restriction are covered by Section VII.

## II. Requirements

We expect that readers are a bit familiar with the basics of the parsing theory presented in many nice books – let us mention at least a few of them: [3], [4], [5], [6].

To change the parser efficiently it is good if the changes are only local – i.e., it is not necessary to generate the entire parser again and only a few its parts are affected. If the changes are on-line, the computation, which has been done, must be preserved. On condition that we use a pushdown automaton, it involves at least the current state and all symbols on the stack. More precisely, it should be possible to get from the current parsing configuration to some final configuration or to other parser-changing configuration.

Therefore, we need a parser structure that is easily modifiable and computation-preserving. Such a requirement holds for structures like a tree or a forest. Well known parsing techniques like LR(k) [7], SLR(k) [8], LALR(k) cannot be used, as even a small change in the grammar may induce large changes in the parser. Other techniques like LL(k) [9], [10] and SLL(k) are of a very limited use: Their structure is a forest, but they are too restrictive in the form of the trees – they can fork only in their root nodes. Restrictions on the parser changes would be too strong, or productions (and the parser structure) would have to be adapted to productions with at most two symbols on their right-hand side (to support more changes). Such grammars are harder to read (understanding them requires more effort and thus using them may induce more practical troubles), so it is better if the system supports productions written "naturally" (i.e., in the way designers think). Therefore, we do not want to be so restrictive in the grammar design.

Ch(k) grammars proposed by Nijholt and Soisalon-Soininen in [11] are quite promising in this sense. Produc-

 ISBN: 978-1-61208-109-0 

tions for every nonterminal may create arbitrary trees. There are, however, two issues:

- re-computation of lookaheads is quite complex for $k > 1$,
- left recursion is not supported. (A production is called left recursive if it is possible to derive a string of symbols from its right-hand side, which starts with the symbol from its left-hand side. A grammar is called left recursive if it has left-recursive productions. Left recursion is often used to describe lists or arithmetic expressions. It is possible to describe them without left recursion [12] but it is less comfortable.)

The tree/forest structure and support for handling left recursion are met by kind grammars and kind parsers [13], [2]. We will describe their structure, show the local character of changes, and discuss the limitation of input language restrictions and modifications.

## III. Structure of Kind Parser

We will explain the structure and behaviour of kind parsers via the "transformation" of a kind grammar to a kind parser and a few examples. In our examples a grammar $G_{SE}$ = (N,Σ,P,S) of a simple arithmetic expression will be used.

*Example 3.1 (Simple arithmetic expression):*

$\mathrm{N} = \{S, E, T, F\}$

$\Sigma = \{id, num, (, ), +, *\}$

$$\mathrm{P} = \left\{ \begin{array}{l} S \to E, \\ E \to E + T, E \to T, \\ T \to T * F, T \to F, \\ F \to id, F \to num, F \to (E) \end{array} \right\}$$

$\mathrm{S} = S$

In one case we will need a set of productions describing a simple command.

*Example 3.2 (Productions of a simple command):*

cmd → **begin** cmds **end**
cmd → id **:=** E
cmd → id ( E )
cmd → **read** id
cmd → **write** E
cmd → **write** str
cmd → **while** cond **do** cmd

Let us start with dividing the productions into groups followed by their simple changes and redrawings. First, the productions are grouped according to the nonterminal on their left-hand side and according to their left recursiveness (Table I).

Then a special symbol is put to the end of each production that denotes the end of a production. These special symbols will be used to handle productions that are prefixes of some other productions and moreover to simplify the parsing process. In many parsing systems the end of a production may start *semantic actions* – activities related to a production or a parsing point. For top-down or semi-top-down parsing it is reasonable to allow semantic actions on arbitrary (or close to arbitrary) places within productions (Table II).

Table I
PRODUCTIONS DEFINING A SIMPLE ARITHMETIC EXPRESSION GROUPED ACCORDING TO THE NONTERMINAL ON THEIR LEFT-HAND SIDE AND RECURSIVENESS

| Productions | |
|---|---|
| without left recursion | with left recursion |
| $S \to E$ | |
| $E \to T$ | $E \to E + T$ |
| $T \to F$ | $T \to T * F$ |
| $F \to id$<br>$F \to num$<br>$F \to (E)$ | |

Table II
SPECIAL-SYMBOL-TERMINATED PRODUCTIONS DEFINING A SIMPLE ARITHMETIC EXPRESSION GROUPED ACCORDING TO THE NONTERMINAL ON THEIR LEFT-HAND SIDE AND RECURSIVENESS

| Productions | |
|---|---|
| without left recursion | with left recursion |
| $S \to E\ \#S_1$ | |
| $E \to T\ \#E_1$ | $E \to E + T\ \#E_2$ |
| $T \to F\ \#T_1$ | $T \to T * F\ \#T_2$ |
| $F \to id\ \#F_1$<br>$F \to num\ \#F_2$<br>$F \to (E)\ \#F_3$ | |

Then we can isolate the left-hand-side nonterminals from the productions. In the left-recursive groups the leading nonterminals can be omitted.

Table III
MODIFIED PRODUCTIONS

| Nonterminal | right-hand sides of productions having | |
|---|---|---|
| | no left recursion | direct left recursion |
| $S$ | $E\ \#S_1$ | |
| $E$ | $T\ \#E_1$ | $+T\ \#E_2$ |
| $T$ | $F\ \#T_1$ | $*F\ \#T_2$ |
| $F$ | $id\ \#F_1$<br>$num\ \#F_2$<br>$(E)\ \#F_3$ | |

After that, the productions are redrawn from the textual form to graphs with edges labelled by symbols (Figure 1).

Next, the productions are reordered so that all productions with the common prefix are together. The common prefixes can be joined to make a single path (Figure 2).

The tree structure is better visible on the example of a simple command (Figure 3).

Finally, the nodes of the trees can be decorated with lookaheads to simplify a traversal along the forest (Figure 4).

## IV. Parsing Process

The work of the kind parser can be described as an input-driven traversal through the production forest – regardless of the representation of the kind parser (which can be in the

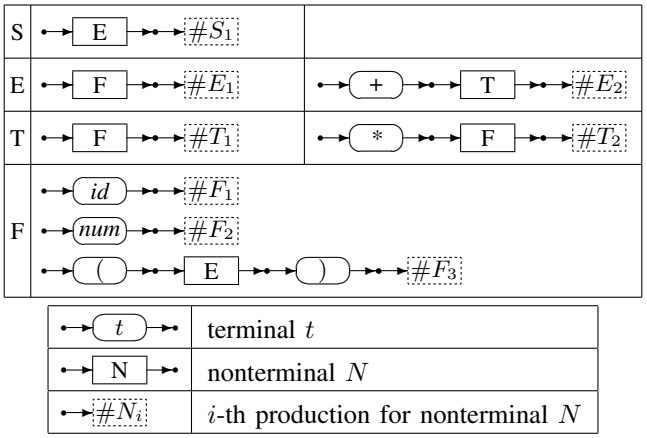



Figure 1. Productions drawn as graph

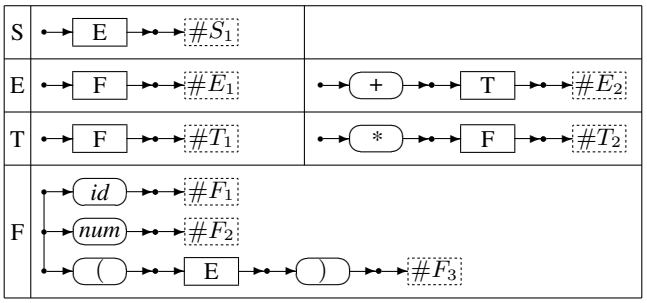



Figure 2. Productions drawn as forest

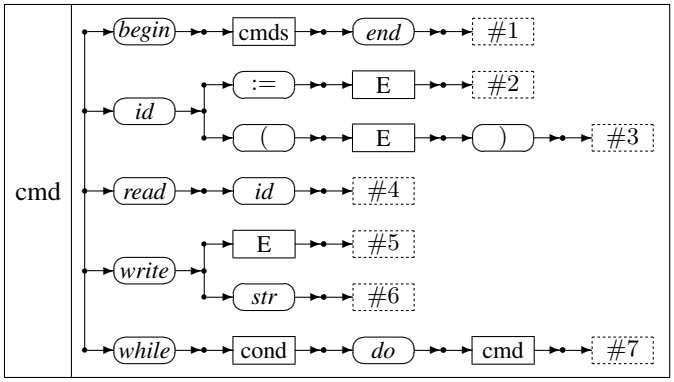



Figure 3. Simple command basic production tree

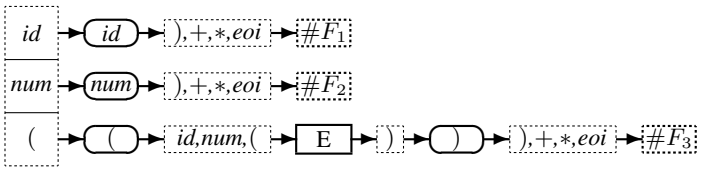



Figure 4. Production tree for nonterminal $F$ with precomputed lookaheads

form of, e.g., a forest, a pushdown automaton, or a set of procedures forming a recursive descent parser).

1) It starts in the root of the non-recursive production tree that is associated with the starting symbol and the pushdown is in its initial setting (e.g., it contains the initial stack symbol only).
2) The lookahead is compared with the lookahead decoration of the node.
3) If there is a path corresponding to the lookahead, the corresponding edge is entered; if no such edge exists, the parsing ends with rejection of the input (or error handling is started).
4) The label type of the current edge is checked:
   a) **terminal**: The input is read and matched with the label – if they do not correspond, the input is rejected and a syntax error is reported. If full lookahead handling is used, the check is redundant. However, in case of lazy lookahead handling (lookaheads are computed for branching nodes only), it can be a necessity.
   b) **nonterminal**: The current position in the forest is put on the stack, the root of the non-recursive production tree is entered.
   c) **production**: The lookahead is checked whether it conforms with the lookahead of the root of the corresponding recursive tree. If it does, the parsing continues here. If not, the Follow set of the given nonterminal (covering only non-recursive cases) is checked. If the current lookahead does not fit, the input is rejected or a syntax error handling procedure is started. If there is a stored position on the stack, it is retrieved and the parsing continues from that parsing position. If the entire input has been read, it is accepted, otherwise only the read part of the input conforms to the input language.

The parsing process of a kind pushdown automaton is described in [14] or [2].

## V. Parser Extension

A kind parser extension is quite simple: New productions are inserted into production trees in such a way that new paths are created there. It holds that all existing nodes and edges of the trees are preserved, only new edges and nodes are added. If new nonterminals are introduced, new production trees occur. Every reference to a node or an edge from the parsing stack existing before the extension is available and usable also after the extension. Every input acceptable without the extension is accepted the same way also after the extension. Hence on-line extensions of kind parsers are safe.

On-line language extensions can be of different extent (their validity may have a different span):

- permanent – from the extension forever;
- temporal – from a given point to another given point

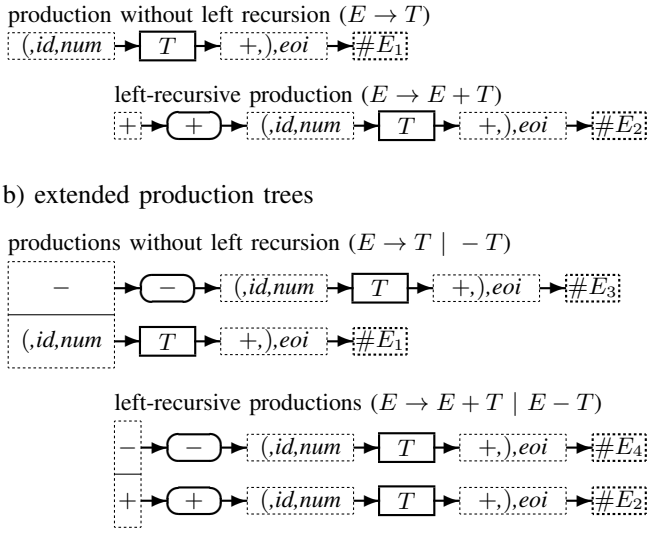



Figure 5. An extension of production trees for nonterminal $E$ with productions $E \to -T$ and $E \to E - T$

(the validity can be, e.g., limited to a language construct);

- semi-permanent – from the extension to a parser restart.

Similarly, extension definitions may have different origin: They are either pre-defined in the language definition, or even part of the input text that is being parsed (or derived from it). A request to switch to the extended language can also be part of the input text or it is an asynchronous event generated by the application that uses the parser. The same holds for parsed language restrictions and modifications discussed below.

The design of extensible grammars may differ from the usual grammar design. It can be advantageous for later extensions to handle also grammars that are not reduced (according to, e.g., [15, p. 273] a context-free grammar $G$ is said to be *reduced* if it contains neither inaccessible nor useless nonterminals; simply said, all symbols must be reachable from the starting symbol and it must be possible to rewrite all nonterminals to a terminal string) – see Example 5.1.

*Example 5.1 (Possible use of a non-reduced grammar):* Let us have a language $L = \{a^n c b^n e a^m c b^m | m, n \geq 0\} \cup \{a^n d b^n e a^m d b^m | m, n \geq 0\}$. The language can be generated by the semantic grammar

$$G = \left( \begin{array}{lcl} N & = & \{S, A, B\} \\ \Sigma & = & \{a, b, c, d, e\}, \\ \Phi & = & \{\sigma, \tau\}, \\ S, & & \\ P & = & \left\{ \begin{array}{l} S \to AeB \\ A \to aAb | c\sigma | d\tau \\ B \to aBb \end{array} \right\} \end{array} \right),$$

where semantic action $\sigma$ adds production $B \to c$ and semantic action $\tau$ adds production $B \to d$.

Extensions also give us an elegant and clear way to express some grammar constructs or even a possibility to express context conditions at the level of the syntax analysis and still use a parser based on lookahead automata and context-free grammars.

## VI. Output

There are several types of output that parsers can produce. The simplest one has mainly theoretical value – parsers just return a logical value stating whether the input word belongs to the accepted language or not.

Practical parsers produce an output that can be further used. One option is a parsing (derivation) tree or information usable for its construction – the sequence of used productions (e.g., the sequence of visited end-of-production edges of our structure).

Another possible output type is a sequence of extra symbols – semantic actions. Usually each semantic action corresponds to some activity of the application using the parser. A special case of such activity can be an extension, a reduction, or even a general modification of the parser.

Parsers can be extended to be transducers. They can transform a message in the input language to another message in the output language. In such a case another type of symbols called *output symbols* or *output terminals* are used. This kind of the parser output can correspond to SAX events known from XML processing.

## VII. Parsed Language Restriction

Having an extensible tool is good. Having an opportunity to remove unnecessary constructs gives some further advantages.

In contrast to a language extension (which is safe for kind parsing), a language restriction requires to be done with some care. For an off-line language restriction usual checks performed for parser construction are sufficient. If we want to reduce the language on-line, we must be more careful, as the already parsed part of the input produced some results that should be preserved for the rest of the parsing process.

It can happen (if we are not careful enough) that a too strong restriction (removal of some parts of the parsing forest) may affect already made computations or even the opportunity to finish the computation itself successfully.

Therefore, some rules must be set, which would guarantee that even after a parser restriction the computation can be finished for a proper input (and the input accepted). We can consider these safety levels:

1) no checking;
2) postponed checking – non-correctness is detected during lookahead recomputation or even during parsing (in case of lazy lookahead handling or when a deleted symbol is popped from the stack);

 ISBN: 978-1-61208-109-0 

3) a sequence of steps leading (for proper input) to the next syntax change or to an accepting situation must be preserved; (We use a situation instead of a configuration, as in the parsing theory a configuration means a triple (state, stack, unread part of input));
4) a sequence of steps leading (for proper input) to the final configuration must be preserved;

We can also check whether the resulting grammar is reduced or not.

The real complexity of checks of resulting grammars depends not only on the grammar size, but also on the current parser stack depth (all the symbols on the stack should be checked). The number of different stack symbols is limited by the grammar, so we can bind the complexity of the checks to the grammar size.

A tool, built according to the theory summarised in this paper, is available in [16] or [17]. It is an implementation of a modifiable kind parser that manages to extend or modify the input language at parse time, both permanently and temporarily. As a demonstration of its power, see Example 7.1. It makes possible to start a program in one language, then switch to another, go back to the first one, and so on. The whole source can be processed in a single pass.

*Example 7.1 (Two languages in one source):*

```
paslike

var A : array[1..10] of integer;
    n : integer;

{Quicksort procedure}
procedure QS (start, stop : integer);

var P, pom : integer;
    i, j : integer;

begin

  if  start < stop  then
    begin

      P:=(A[start] + A[stop]) div 2;
      i:=start;
      j:=stop;

      while i <= j do
        begin
          clike
            /* find items to change */
            while (A[i] < P) ++i;
            while (A[j] > P) --j;
            if (i <= j){
                pom=A[i]; A[i]=A[j]; A[j]=pom;
                ++i;
                --j;
             }
          finish
        end;

      QS(start,j);
      QS(i, stop)

    end;
end;

clike

int main(int argc,char **argv) {
        fill(A);
        QS(1, 10);
        present(A);
        return 0;
}

finish

finish
```

## VIII. Conclusion

Kind parsing presented in this paper can be – as an adaptive technology – a very useful tool for adaptive system development. It is an efficient and easy to understand parsing technique that supports on-line and off-line adaptation (extensions, reductions, general changes) of the accepted language.

The on-line changes can be induced by information contained in the input that is being parsed (processed as a semantic action) or even by an asynchronous parser changing event generated by the application using the parser, i.e., the application can change the behaviour of the underlying parser processing a data stream.

The theory introduced in this paper was practically verified by implementing tools.

We suppose that such adaptive parsers can be used at least in these cases:

- parsing/compiling extensible languages,
- (adaptive) front-end gates [18],
- input parts of adaptive systems controlled from outside (other applications or adaministrators),
- input parts of (self-)adaptive systems controlled by the input text/data,
- input parts of self-adaptive systems controlled by the application logic.

The only restriction is that the cooperating parts of the system influenced by the adaptation must be adapted correspondingly.

## Acknowledgment

This paper was partially supported by the Czech Science Foundation by the grant number 201/09/0983.

## References

[1] M. Žemlička and J. Král, "Run-time extensible (semi-)top-down parser," in *Proceedings of the 2nd International Workshop on Text, Speech and Dialogue (TSD-99)*, ser. LNAI, V. Matoušek, P. Mautner, J. Ocelíková, and P. Sojka, Eds., vol. 1692. Berlin: Springer, Sep. 13–17 1999, pp.

121–126. [Online]. Available: http://dx.doi.org/10.1007/3-540-48239-3_22 [Last visited: August 31, 2010]

[2] M. Žemlička, "Introduction to kind parsing," Ph.D. dissertation, Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic, Jul. 2006. [Online]. Available: http://www.ksi.mff.cuni.cz/~zemlicka/pdf/PrinciplesOfKindParsing.pdf [Last visited: August 31, 2010]

[3] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*. Englewood Cliffs, N.J.: Prentice-Hall, 1972, vol. I.: Parsing.

[4] A. V. Aho, R. Sethi, and J. D. Ullman, *Compilers: Principles, Techniques, and Tools*, ser. World Students Series Edition. Addison-Wesley, 1986, vol. 10194.

[5] A. Meduna, *Automata and Languages: Theory and Applications*. London: Springer, 2000.

[6] D. Grune and C. J. H. Jacobs, *Parsing Techniques*, 2nd ed., ser. Monographs in Computer Science. New York, USA: Springer, 2008. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-68954-8 [Last visited: August 31, 2010]

[7] D. E. Knuth, "On the translation of languages from left to right," *Information and Control*, vol. 8, no. 6, pp. 607–639, 1965. [Online]. Available: http://dx.doi.org/10.1016/S0019-9958(65)90426-2 [Last visited: August 31, 2010]

[8] F. DeRemer, "Simple LR(k) grammars." *Commun. ACM*, vol. 14, no. 7, pp. 453–460, 1971. [Online]. Available: http://doi.acm.org/10.1145/362619.362625 [Last visited: August 31, 2010]

[9] P. M. Lewis, II and R. E. Stearns, "Syntax directed transduction," in *Switching and Automata Theory*. IEEE, 1966, pp. 21–35. [Online]. Available: http://dx.doi.org/10.1109/SWAT.1966.26 [Last visited: August 31, 2010]

[10] D. J. Rosenkrantz and R. E. Stearns, "Properties of deterministic top-down grammars," *Information and Control*, vol. 17, no. 3, pp. 226–256, 1970. [Online]. Available: http://dx.doi.org/10.1016/S0019-9958(70)90446-8 [Last visited: August 31, 2010]

[11] A. Nijholt and E. Soisalon-Soininen, "Ch(k) grammars: A characterization of LL(k) languages." in *MFCS*, ser. Lecture Notes in Computer Science, J. Becvár, Ed., vol. 74. Springer, 1979, pp. 390–397. [Online]. Available: http://dx.doi.org/10.1007/3-540-09526-8_38 [Last visited: August 31, 2010]

[12] E. Soisalon-Soininen and E. Ukkonen, "A method for transforming grammars into $LL(k)$ form," *Acta Informatica*, vol. 12, pp. 339–369, 1970. [Online]. Available: http://dx.doi.org/10.1007/BF00268320 [Last visited: August 31, 2010]

[13] M. Žemlička and J. Král, "Run-time extensible deterministic top-down parsing," *Grammars*, vol. 2, no. 3, pp. 283–293, 1999. [Online]. Available: http://dx.doi.org/10.1023/A:1009914518458 [Last visited: August 31, 2010]

[14] J. Král and M. Žemlička, "Semi-top-down syntax analysis." in *Grammars and Automata for String Processing*, ser. Topics in Computer Mathematics, C. Martín-Vide and V. Mitrana, Eds., vol. 9. London: Taylor and Francis, 2003, pp. 77–90.

[15] R. Wilhelm and D. Maurer, *Compiler Design*, ser. International Computer Science. Wokingham, England: Addison-Wesley, 1995, vol. 42290.

[16] P. Šašek, "Rozšiřování syntaxe za běhu (in Czech: Run-time syntax extensions)," Master's thesis, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, 2007.

[17] P. Šašek, ExTra - Extensible Transducer. [Online]. Available: http://www.ksi.mff.cuni.cz/~zemlicka/projects/download/ExTra/ [Last visited: August 31, 2010]

[18] J. Král and M. Žemlička, "Software architecture for evolving environment," in *Software Technology and Engineering Practice*, K. Kontogiannis, Y. Zou, and M. D. Penta, Eds. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 49–58. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/STEP.2005.25 [Last visited: August 31, 2010]

 ISBN: 978-1-61208-109-0 

# Efficiency Testing of Self-adapting Systems by Learning of Event Sequences

Jonathan Hudson, Jörg Denzinger
*Department of Computer Science, University of Calgary*
*Calgary, Canada*
*Email: {hudsonj, denzinge}@cpsc.ucalgary.ca*

Holger Kasinger, Bernhard Bauer
*Department of Computer Science, University of Augsburg*
*Augsburg, Germany*
*Email: {kasinger, bauer}@informatik.uni-augsburg.de*

***Abstract*—Adding self-adaptation as a property to systems aims at improving the efficiency of this system. But there is always the possibility for adaptations going too far, which is very detrimental to the trust of users into a self-adapting system. In this paper, we present a method for testing the efficiency of a self-adapting system, more precisely the potential for inefficiencies after adaptation has taken place. Our approach is based on learning sequences of events that set the system up so that a second following learned sequence of events is reacted to very inefficiently by the system. We used this approach to evaluate a self-adapting system for solving dynamic pickup and delivery problems and our experiments show that the potential inefficiencies due to self-adaptation are smaller than the inefficiencies that the non-adapting base variant of the system is creating.**

***Keywords*-testing; learning; dynamic optimization**

## I. Introduction

Quality and thoroughness of testing are important factors for the trust of users into any kind of system, be it physical systems like power plants or cars or pure software systems. Naturally, what constitutes thorough and high quality testing depends very much on the system that is tested and the properties it is tested for. While testing "conventional" systems for their intended behavior essentially boils down to having the time to go through all these behaviors, testing them for "negative" properties, like under no circumstances showing a certain behavior, already is a difficult task and the quality of the testing heavily depends on the human tester and his abilities. For self-adapting systems, this becomes an even more difficult task, since unwanted or bad behavior might only emerge after a series of adaptations of the system. And if such a system consists of several interacting components (*agents*), then a tester faces additionally the danger of unwanted emergent behavior, also called *emergent misbehavior* (see [6]). Expecting every human tester to find crucial problems with self-adapting systems with the potential for emergent behavior is a highly doubtful assumption and does not instill trust in such systems in general.

In this paper, we present an automated approach to test a self-adapting, self-organizing multi-agent system that solves dynamic pickup and delivery problems. A key property for such a system is the efficiency of the delivery behavior and, with regard to its self-adaptation, the potential for (temporary) loss of efficiency due to self-adaptation and change of the environment. Our testing approach is based on learning sequences of events for the tested system to encounter and react to. More precisely, we extend the evolutionary learning approach of [2] to search for two sequences of events that represent tasks and when they are announced to the tested system. One sequence, the set-up, is aimed at having the system adapt itself to it by exposing the system repeatedly to this sequence. After the system is optimally adapted, the break sequence is given to it and the aim of our learning test system is to find two such sequences so that the efficiency achieved by the tested system for the break sequence is much worse after adaptation than without adaptation, providing users with a practical example of how bad a temporary loss of efficiency can get.

We applied our approach to the self-adapting improvement of a system for dynamic pickup and delivery problems based on digital infochemical coordination (see [4]). The self-adaptation is achieved by a so-called advisor that identifies recurring task sequences that are not well handled by the base system, determines how the tasks should be handled and creates exception rules for the transportation agents that achieve the intended solution (see [9]). We used our testing approach also to find problem instances for which the base system of [4] is not very efficient. Our test system found event sequences where the base system's efficiency was on average 3.5 times worse than the optimal solution, whereas the self-adapting variant was only two times worse for the break sequence than without self-adaptation. This is, in our opinion, a rather good result for the self-adapting variant, since for many instances this variant improves the behavior substantially (as documented in [4]).

After this introduction, in Section II we present the general idea of learning of event sequences for testing. Section III introduces our application area and Section IV the system to be tested. Section V instantiates the general idea for the system and Section VI reports on our experimental evaluation. After a view on the related work in Section VII, Section VIII concludes with some ideas for future work.

## II. Testing using Learning of Event Sequences

In this section, we present our general scheme for testing a self-adapting system for potential loss of efficiency due to self-adaptation using learning of event sequences. While

 ISBN: 978-1-61208-109-0 

a self-adapting system naturally does not have to consist of several agents, the system we present in Section IV for instantiating our approach does, so that we assume that our self-adapting system to be tested is a set $A_{tested} = \{Ag_{tested,1}, ..., Ag_{tested,m}\}$ of agents. Our system $A_{tested}$ will work within an environment $Env$. There might be other systems acting in $Env$, either single agents or groups, that interact with $A_{tested}$ and the environment itself might also change. We see the actions of other systems as well as all environmental changes as events that may or may not influence $A_{tested}$. In order to allow for events caused by other systems, for our testing we add a new group of agents $A_{evgen} = \{Ag_{evgen,1}, ..., Ag_{evgen,n}\}$ that are controlled by a learner and that are generating events in the environment for $A_{tested}$ to react to. This general setting is depicted in Figure 1.

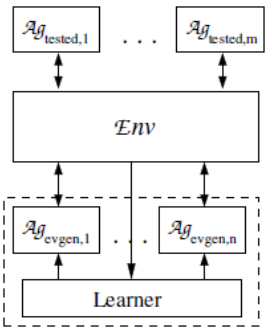



Figure 1. General setting of our approach

Formally, each agent $Ag_{evgen,i}$ creates a sequence of events $((ev_1^i, t_1^i), (ev_2^i, t_2^i), ..., (ev_l^i, t_l^i))$, which along with the reaction of the agents of $A_{tested}$ produce a sequence of environmental states $e_0, e_1, ..., e_x$. This sequence of environmental states is utilized by the machine learner to evaluate the associated sequence of events to find better event sequences.

While several different machine learning techniques can be used to learn event sequences, we will use an evolutionary learning approach as suggested in [2]. Starting with a set of randomly created event sequences for each $Ag_{evgen,i}$, each element (*individual*) is evaluated by running the events in the environment and analyzing the resulting sequence of environmental states using a *fitness function*. Then the best individuals are used to create new individuals using *genetic operators*. The best individuals together with the new ones form a new set (generation) and this process is repeated for a given number of rounds.

## III. Dynamic Pickup and Delivery Problems

The general pickup and delivery problem (PDP, see [7]) is a well-known problem class. Many of its instantiations require solving dynamic instances of this problem and many of those have many instances that allow for a self-adapting system to improve efficiency. There are also several variants of the PDP, for example PDP with time windows, that add additional constraints to the problem. While our testing method can be used for systems for all of these variants, in our experiments we used a system solving the following variant.

As the name suggests, a PDP consists of a sequence of pickup and delivery tasks and in a dynamic PDP these tasks are announced not all at the beginning but over a period of time. The tasks are performed by one or several transportation agents. The variant of PDP we are interested in requires repeatedly solving such sequences of tasks, where a sequence is called a *run instance*, and, in order to have a chance for self-adaptation, we require that run instances have at least several recurring tasks.

More formally, a run instance for a dynamic PDP is a sequence of task-time pairs $((ta_1, t_1), (ta_2, t_2), ..., (ta_k, t_k))$, where each $ta_i$ consists of a pickup location $l_{pick,i}$, a delivery location $l_{del,i}$, and a required capacity $ncap_i$ and $t_i \in Time$, $t_i \leq t_{i+1}$, with $Time$ being the time interval in which the run instance is to be performed. $t_i$ is the time at which $ta_i$ is announced to the transportation agents. A transportation agent $Ag$ has a transport capacity $cap_{Ag}$ and has to perform both the pickup and the subsequent delivery to complete a task.

The solution $sol$ produced by a set of transportation agents $\{Ag_1,...,Ag_m\}$ is represented as

$$sol = ((ta'_1, Ag'_1, t'_1), (ta'_2, Ag'_2, t'_2), ..., (ta'_k, Ag'_k, t'_k))$$

where $ta'_i \in \{ta_1, ..., ta_k\}$, $ta'_i \neq ta'_j$ for all $i \neq j$, $Ag'_i \in \{Ag_1, ..., Ag_m\}$, $t'_i \leq t'_{i+1}$, $t'_i \in Time$. A tuple $(ta'_i, Ag'_i, t'_i)$ means that the pickup of $ta'_i$ was done by $Ag'_i$ at time $t'_i$.

There are many possibilities how to measure the efficiency $eff(sol)$ of the agents when producing $sol$. Examples are distance covered by the agents, time needed to fulfill all tasks, balance among the agents and many more, especially all kinds of combinations. For our variant, we use as efficiency measure the total distance traveled by all agents. There are also many different environments in which agents can work on the tasks. In our variant, we use a simple grid, where each node represents a possible location.

## IV. Solving PDP using Digital Infochemicals and an Advisor

Solving all kinds of dynamic PDP variants is rather difficult, simply because of the dynamic nature that requires to make decisions without really knowing what additional tasks might come up later. Usually, the time frames for a run instance do not allow to create an optimal plan every time a new task is announced (the static PDP problem is already in NP for most variants). As a consequence, many of the systems used to solve dynamic problems mostly ignore the efficiency aspect and concentrate on other useful system

 ISBN: 978-1-61208-109-0 

properties, like robustness against failure, graceful degradation, easy extendibility and general openness of the system. Digital infochemical coordination (see [4]) is a coordination concept for self-organizing multi-agent systems that shows these properties and it has been used as the coordination concept for a system solving the variant of PDP presented in the last section. Essentially, each location in each task is represented by an agent emitting infochemicals that lead the transportation agents to them when a task is announced. The transportation agents also emit infochemicals to coordinate and avoid having all of them moving to the same pickup location. If a location agent has been served, it, again, uses specific infochemicals to inform the transports that they are not needed by it anymore. Due to lack of space and because a deeper understanding of this base system is not necessary for understanding our instantiation of the testing idea from Section II, we will not go into any more detail of this base system. The interested reader should consult [4].

From a practical application perspective, efficiency of a pickup and delivery system cannot be ignored! Therefore, [9] extended the system from [4] to improve efficiency using an additional agent, an *efficiency improvement advisor* $Ag_{EIA}$, that essentially adds to the base system the ability to reflect on the behavior shown by the transportation agents and to adapt this behavior after having seen really inefficient behavior that is likely to happen again in future run instances. Since $Ag_{EIA}$ is part of the system, the system as a whole is self-adapting.

The advisor works as follows: when the agents have dealt with a run instance, they go back to their depot at which the advisor is located and report their local history to it. The advisor uses this data to compile a global history for the whole system over a given number of run instances. Using this global history, $Ag_{EIA}$ identifies a sequence $(ta_1^{rec}, ..., ta_l^{rec})$ of recurring tasks by clustering all tasks from the run instances using Sequential Leader Clustering (see [3]). Every cluster with a size slightly smaller or equal to the number of run instances indicates a recurring task. The clustering makes use of a similarity measure for tasks, but for our testing this measure is not of interest. Then $Ag_{EIA}$ computes an optimal (or at least very good) solution for the sequence of recurring tasks according to the efficiency measure $eff$ using an optimization algorithm for the static variant of our PDP (in our case, we used a genetic algorithm that uses or-trees as basis for the genetic operators). If the optimal solution is much better than what the agents generated in the last run instance, $Ag_{EIA}$ starts creating advice for the agents.

The advice consists of so-called *ignore rules* that are iteratively produced by $Ag_{EIA}$. It compares the produced solution with the optimal one and identifies the first position where the two solutions are not identical (with regard to task or agent). It then creates an ignore rule for the agent that performed the task in the produced solution that essentially has as condition the task (its pickup location and the needed capacity) and as action the advice to the agent to *not* do this task. Since the base system is self-organizing, another agent will pick up the task now ignored by the previous (and from the perspective of efficiency wrong) agent. After an ignore rule was given to an agent, the advisor waits until after the next run instance to see if now the produced solution is near enough to the efficiency of the optimal one. If not, the production step for an ignore rule is repeated.

By repeating the clustering every time new data is available allows for dealing with a change of recurring tasks over time. Theoretically a big change of tasks between two run instances could result in a big (temporary) loss of efficiency for the second run instance. Getting some practical idea how bad this potential loss can get can improve a user's trust into the system dramatically, which is what our testing system in the next section is aimed at producing.

## V. Instantiating our Testing Approach

There are several potential sources for a system as described in the last section to produce rather inefficient solutions to run instances and thus several sources for distrust in the system. We are interested in the effects of the advisor and especially the potential for overadaptation and the consequent loss of efficiency when the system encounters a run instance with a unique set of tasks after adapting to a different recurring set of tasks.

When instantiating the general idea from Section II to create a test system for the self-adapting system presented in Section IV, we followed the usual approach of human testers to concentrate on the test goal and to eliminate influences that are not in the test goal. So, in order to speed up adaptation in our test system, there are no non-recurring tasks in the setup run instance. The test system concentrates of finding a setup sequence of tasks (events), that the systems repeatedly encounters and adapts to, along with a second sequence, which is encountered after the setup adaptations. And the test goal is to find such a pair of sequences for which the efficiency for the second sequence after adaptation is much worse than without adaptation taking place. The transportation agents (and the advisor) form the set $A_{tested}$. In [4] and [9], agents at the appropriate locations for a task announce the tasks to the tested agents, so that we have indeed a set of event generating agents. But for the number of tasks we use in our experiments, we have a lot of locations that are not used by them. Therefore we use run instances in our individuals and create them centrally in our learner.

More precisely, an individual of our evolutionary learner is a pair $(es_{setup}, es_{break})$, where each of the two run instances is a sequence of task-time pairs. As evolutionary operators we use the usual single point mutation and crossover on lists by first selecting a position $o$ in either $es_{setup}$ or $es_{break}$. A mutation then takes the run instance $((ta_1, t_1), ..., (ta_l, t_l))$ and creates

$$((ta_1, t_1), ..., (ta_o^{new}, t_o^{new}), ..., (ta_l, t_l))$$

where either $ta_o^{new}$ is a different task than $ta_o$ or $t_o^{new} \neq t_o$.

Crossover takes two run instances $((ta_1^1, t_1^1), ..., (ta_l^1, t_l^1))$ and $((ta_1^2, t_1^2), ..., (ta_l^2, t_l^2))$, and creates a new instance

$$((ta_1^{new}, t_1^{new}), ..., (ta_l^{new}, t_l^{new}))$$

where each $(ta_i^{new}, t_i^{new})$ is either equal to $(ta_i^1, t_i^1)$ or to $(ta_i^2, t_i^2)$. Then the new individual copies the not selected run instance from the first parent pair and adds the newly created run instance as the other element of the new pair.

We also added a so-called targeted operator, which is a twin point mutation operator that attempts to create similar tasks between the two run instances of the individual to allow the ignore rules created for adaptation to the setup instance to negatively impact the break instance. This is achieved by aligning a task-time pair between the instances. If the individual $(es_{setup}, es_{break})$ is

$$(((ta_{1,1}, t_{1,1}), ..., (ta_{1,l}, t_{1,l})), ((ta_{2,1}, t_{2,1}), ..., (ta_{2,l}, t_{2,l})))$$

then the targeted twin point mutation operator selects a position $i$ and creates the new individual

$$(((ta_{1,1}, t_{1,1}), ..., (ta_{1,i}^{new}, t_{1,i}^{new}), ..., (ta_{1,l}, t_{1,l})),$$
$$((ta_{2,1}, t_{2,1}), ..., (ta_{2,i}^{new}, t_{2,i}^{new}), ..., (ta_{2,l}, t_{2,l})))$$

where $ta_{1,i}^{new}$ and $ta_{2,i}^{new}$ are identical tasks and $t_{1,i}^{new} = t + \epsilon_1$ and $t_{2,i}^{new} = t + \epsilon_2$ for randomly chosen $t \in Time$ and small numbers $\epsilon_1$ and $\epsilon_2$.

While an obvious fitness measure for an individual $(es_{setup}, es_{break})$ would be to simply compute the difference in efficiency between the created solution for $es_{break}$ with adaptation ($eff(sol_{+ad}(es_{break}))$) and without adaptation (($eff(sol_{-ad}(es_{break}))$), our initial experiments showed that the learner needed some more "advice" to create the individuals we wanted quickly. More precisely, the learner had problems due to many early individuals where no adaptation took place and due to individuals where $sol_{+ad}(es_{break})$ was too near to the optimal solution. Both types of individuals clearly are not of interest for our test goal and therefore we created a fitness measure punishing them:

$$fit_{+ad}((es_{setup}, es_{break})) = \frac{pract(es_{break}) + theo(es_{break}) + adapt(es_{setup})}{eff(sol_{opt}(es_{break}))}$$

with

$$pract(es_{break}) = max[(eff(sol_{+ad}(es_{break})) - eff(sol_{-ad}(es_{break}))) * w_{pract}, 0]$$
$$theo(es_{break}) = max[(eff(sol_{+ad}(es_{break})) - eff(sol_{opt}(es_{break}))) * w_{theo}, 0]$$
$$adapt(es_{setup}) = max[(eff(sol_{-ad}(es_{setup})) - eff(sol_{+ad}(es_{setup}))) * w_{adapt}, 0]$$

and $sol_{opt}(es_{break})$ being the optimal solution for the break run instance.

We account for the difference of the solution quality produced for $es_{break}$ before and after adaption weighted by a parameter $w_{pract}$. We also take account of the difference between the emergent solution for $es_{break}$ after adaptation and the optimal solution weighted by a parameter $w_{theo}$ measuring how far the solution is from the theoretical optimum, along with taking account of the difference between the solution produced for $es_{setup}$ before and after advice, using $w_{adapt}$ as parameter for its importance.

The learning testing system for the self-adapting system for PDP as described above can be easily modified for other testing goals around efficiency of a system. For example, the efficiency of the underlying self-organizing base system can be tested using only one run instance as individual (essentially just having $es_{break}$), not using the targeted twin point mutation operator and using as a fitness function

$$fit_{base}(es_{break}) = \frac{eff(sol_{-ad}(es_{break}))}{eff(sol_{opt}(es_{break}))}.$$

In the next section, we will not only report on our experiments testing the self-adapting system for PDP, we will also provide results on how bad the efficiency of the base system can be compared to an optimal solution, to put our results for the self-adapting system into perspective.

## VI. Experimental Results

In this section, we describe several experiments performed using our test system to evaluate the self-adapting system for PDP from Section IV. First we present the general settings for the experiments and then the results of our test system for the settings. To put these results in perspective, we also used the variant of the test system described in the last section for the base system for PDP.

### A. Experimental Settings

Each experiment used a $10 \times 10$ grid with a depot in the middle. We used the settings for the various parameters of the base system and advised variant reported in [9]. Our testing used the following percentages for a new population to be created by the evolutionary operators: 10 percent best survived, 30 percent generated using crossover and 60 percent via mutation, 30 for each kind of mutation.

Every experimental series consisted of 5 runs of the testing system due to the random effects of the evolutionary learning process. In all experiments we used two transportation agents, $Ag_1$ and $Ag_2$, and 2, 4, 6, 8 and 10 tasks. This limitation to two agents is because within the chosen range of task sizes the addition of other agents was unnecessary so that also self-adaptation would not be necessary. Also, from a testing perspective we are interested in small examples that show a problem. $Time$ was an integer interval of $[0, 50]$ for 2 tasks, $[0, 100]$ for 4 tasks, $[0, 150]$ for 6 tasks, $[0, 200]$ for 8 tasks and $[0, 250]$ for 10 tasks. For each number of tasks we performed one run to get an idea after what generation no improvement seemed to occur anymore and we used this number to limit the run length of the other test runs. For the provided results we indicate both the average efficiency loss and the maximal loss among the 5 runs.

 ISBN: 978-1-61208-109-0 

### B. Quantitative Results

Table I summarizes our experimental results for testing the self-adapting system for the PDP as set up in the last subsection. For the fitness function we used weights of $w_{pract} = 25$, $w_{adapt} = 5$, and $w_{theo} = 1$. This means that the primary component is how much worse the break run instance is solved after adaptation, compared to the efficiency the system shows for the break instance without the adaptation. The other components are there to make sure that the setup instance really leads to an adaptation and that the break instances are solved badly, as described in the last section.

| Tasks | Generations | Average | Maximum |
|---|---|---|---|
| 2 | 35 | 1.7 | 1.9 |
| 4 | 70 | 1.8 | 1.9 |
| 6 | 105 | 2.0 | 2.1 |
| 8 | 140 | 1.9 | 2.1 |
| 10 | 175 | 2.0 | 2.0 |

Table I
EFFICIENCY LOSS RESULTS FOR SELF-ADAPTING SYSTEM

Table I shows the average efficiency loss due to the adaptation is around a factor of 2 and the worst found examples are also very near to that factor[1]. This means that in the rather extreme situation where a total change of the tasks to fulfill can happen (which is usually not the case in the scenarios for which the advisor was developed) the system result is only two times worse than without the advisor. With regard to the efficiency of our test system itself, the 5 runs accounted for in the 10 task entry of the table took 19.4 hours to complete.

Before we look more closely at one of the runs from Table I to see what causes the loss of efficiency, we will first look at the results of our testing system when modified to evaluate the efficiency of the base self-organizing system. Table II presents the results of our test system for the same numbers of tasks and the same grid setting as for Table I. As can be seen, the worst event sequences found by our test system are clearly worse than the efficiency loss potential found for the self-adapting system. One of the goals of the particular self-adapting system we are testing was to keep the strengths of the base system, especially the self-organization ability, so that big changes in events from one run instance to the next would have no big impact. Tables I and II show that this goal was indeed achieved.

### C. A "bad" problem instance

Figures 2 and 3 visualize one of the examples with 4 tasks. The setup run instance is

((F,K),19),((L,H),35),((C,G),74),((J,A),75)

[1]As already stated, the efficiency loss is computed as the efficiency of the solution for $es_{break}$ after adaptation divided by the efficiency of the base system without adaptation for $es_{break}$.

| Tasks | Generations | Average | Maximum |
|---|---|---|---|
| 2 | 40 | 2.6 | 2.8 |
| 4 | 80 | 3.4 | 4.9 |
| 6 | 120 | 3.4 | 3.7 |
| 8 | 160 | 3.6 | 4.5 |
| 10 | 200 | 3.6 | 4.5 |

Table II
EFFICIENCY PROBLEMS OF BASE SYSTEM ($fit_{base}$)

and the break instance is

((D,E),18),((F,I),38),((B,D),55),((J,F),78).

Without the advisor, the system solves the setup instance by having $Ag_1$ (indicated by the dashed lines) fulfill task ((F,K),19), starting to respond to task ((C,G),74) but then switching to task ((J,A),75). $Ag_2$ does ((L,H),35), starts to respond to ((C,G),74), switches to start to respond to ((J,A),75) and then switches back to fulfilling ((C,G),74). The advisor realizes that one agent should do all tasks and creates exception rules for $Ag_1$ to ignore tasks starting at F after time 19, L after time 35, C after time 74 and J after time 75. As the bottom left part of Figure 2 shows, this does not result in a perfect solution after the advice, since $Ag_2$ fulfills ((F,K),19) then ((L,H),35), then starts to fulfill ((C,G),74), but switches to ((J,A),75) and then comes back to fulfilling ((C,G),74). This shows a limitation of the existing ignore exception rules provided by the advisor since it cannot enforce a time for the agents to complete tasks in.

Without having the advisor, the break run instance is solved by the system by having $Ag_1$ fulfill (((D,E),18), then ((F,I),38) and then ((J,F),78). $Ag_2$ fulfills ((B,D),55). As with so many of the instances for a system with two agents, the optimal solution would be to have one agent do all tasks. After having adapted to the setup instance, the system solves the break instance in the following manner. $Ag_1$ fulfills task ((D,E),18), ignores ((F,I),38) because of the exception rule, partially responds to ((B,D),55), discards this task and then ignores ((J,F),78) due to the other exception rule. This means that $Ag_2$ fulfills first ((F,I),38), then does ((B,D),55), and then performs ((J,F),78).

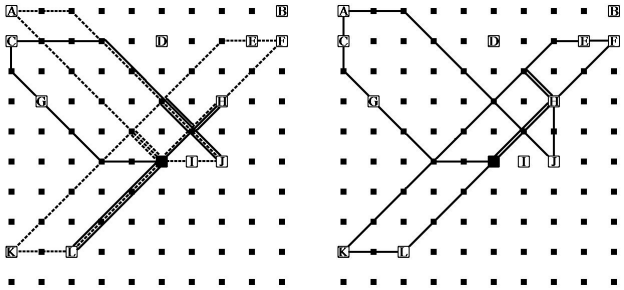



Figure 2. Setup without Advice (Right) vs. Setup with Advice (Left)

This example shows the importance of having targeted operators to bring knowledge about the tested system into the

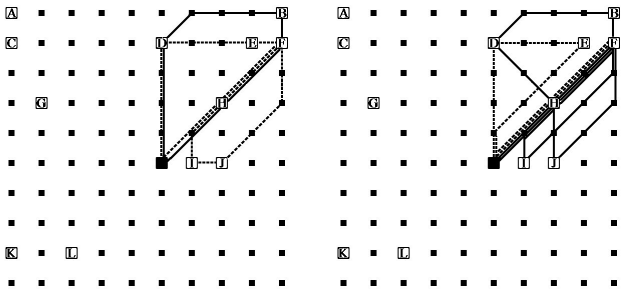


Figure 3. Break without Advice (Right) vs. Break with Advice (Left)

learning process. The twin point mutation operator connects tasks in the setup and the break instance to create events that trigger the exception rules, which may then result in unwanted behavior. These unintended consequences are the danger of an advisor and our test system gives us an idea of the potential inefficiencies produced.

## VII. Related work

The use of learning/self-adapting systems to test systems for certain properties/test goals has become a very active research area over the past years, although not under the terms "learning" or "self-adaptation" (many use "search based" instead). Many of the evolutionary approaches for testing reported at [8] are, in fact, learning systems. For example, [1] evolves a schedule of given events for a scheduler (resp. an executable model of it) with the goal to find times for the announcement of the given events that lead to infeasible schedules. [1] comes nearest to our approach, but the tasks have to be given, not learned, and the tested system is a single agent, not a group.

With regard to testing self-adapting or even just self-organizing systems, this topic has not drawn a lot of attention, so far. Naturally, for testing the wanted behavior of such systems, standard testing methods can be and are used. But the added difficulty to "negative" testing has not been the focus of research beyond the already mentioned works.

## VIII. Conclusion and Future Work

We presented a method to test self-adapting systems for adaptations that result in a loss of efficiency and for the tasks that are solved less efficiently. By learning sequences of events that set up the system so that a follow-up sequence is badly solved, our method provides users of self-adapting systems with an idea what can go wrong and allows them to evaluate this risk compared to the gains the self-adapting system is providing. This aims at increasing the trust into the system. Naturally, the developers of the system also can use the found sequences to improve their system.

We used a system based on our method to test a self-adapting, self-organizing multi-agent system for one variant of dynamic pickup and delivery problems. Our experiments showed that while our system was only able to find event sequences for the self-adapting system that were around 2 times worse than what the system without the self-adaptation would have achieved, for this base system without adaptation a variant of our test system was able to find on average event sequences that it solved around 3.5 times worse than would be the optimum solution. In our opinion, this strengthens the claim of the self-adapting system developers that their adaptation approach is very targeted to situations in which the base system is bad and represents only a minimal intrusion into the base mechanism.

Naturally, like all testing, our method cannot guarantee to find the worst event sequence there is for the tested system. But, compared to a human tester, it has no bad days and works always on a consistent level, especially if a test consists of several runs of the system as in our experiments. And our system does not retire and leave a company with novice testers.

There are several directions for future research. The developers of the self-adapting system we used have ideas for additional types of exception rules that are more invasive into the base system than the ignore rules (see [5]). Testing these new rules should provide an idea if this increases the potential for inefficiency. Along the development and integration of these new exception rules, our test system can also be used to realize some kind of test-driven development for self-adapting systems. Finally, we plan to use our method to develop test systems for other self-adapting systems for other applications.

## References

[1] L. Briand, Y. Labiche, and M. Shousha: Using Genetic Algorithms for Early Schedulability Analysis and Stress Testing in Real-Time Systems, Genetic Programming and Evolvable Machines 7(2), 2006, pp. 145–170.

[2] B. Chan, J. Denzinger, D. Gates, K. Loose, and J. Buchanan: Evolutionary behavior testing of commercial computer games, Proc. CEC 2004, Portland, 2004, pp. 125–132.

[3] J.A. Hartigan: Clustering Algorithms, John Wiley and Sons, 1975.

[4] H. Kasinger, B. Bauer, and J. Denzinger: Design Pattern for Self-Organizing Emergent Systems Based on Digital Infochemicals, Proc. EASe 2009, San Francisco, 2009, pp. 45–55.

[5] H. Kasinger, B. Bauer, J. Denzinger, and T. Holvoet: Adapting Environment-Mediated Self-Organizing Emergent Systems by Exception Rules, Proc. SOAR 2010, Washington, 2010.

[6] J.C. Mogul: Emergent (mis)behavior vs. complex software systems, SIGOPS Operating Systems Review 40(4), 2006, pp. 293–304.

[7] M.W.P. Savelsbergh and M. Sol: The General Pickup and Delivery Problem, Transp. Science 30, 1995, pp. 17-29.

[8] SEBASE: Software Engineering By Automated SEarch Repository, http://www.sebase.org/sbse/publications/, as seen on Jun. 18, 2010.

[9] J.P. Steghöfer, J. Denzinger, H. Kasinger, and B. Bauer: Improving the Efficiency of Self-Organizing Emergent Systems by an Advisor, Proc. EASe 2010, Oxford, 2010, pp. 63–72.

 ISBN: 978-1-61208-109-0 