ADAPTIVE 2015

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

ISBN: 978-1-61208-391-9

March 22 - 27, 2015

Nice, France

ADAPTIVE 2015 Editors

Mark J. Balas, Embry-Riddle Aeronautical University - Daytona Beach, USA

Dirk Malzahn, Dirk Malzahn Ltd. / HfH University, Germany

ADAPTIVE 2015

Forward

The Seventh International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2015), held between March 22-27, 2015 in Nice, France, continued a series of events targeting 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.

The conference had the following tracks:

- Self-adaptation
- Adaptive applications
- Adaptivity in robot systems
- Fundamentals and design of adaptive systems

Similar to the previous edition, this event attracted excellent contributions and active participation from all over the world. We were very pleased to receive top quality contributions.

We take here the opportunity to warmly thank all the members of the ADAPTIVE 2015 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to ADAPTIVE 2015. 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 also gratefully thank the members of the ADAPTIVE 2015 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope ADAPTIVE 2015 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of adaptive and self-adaptive systems and applications. We also hope that Nice, France provided a pleasant

environment during the conference and everyone saved some time to enjoy the charm of the city.

ADAPTIVE 2015 Chairs

ADAPTIVE 2015 Advisory Chairs

Radu Calinescu, University of York, UK
Thomas H. Morris, Mississippi State University, USA
Serge Kernbach, CYBERTRONICA RESEARCH-Stuttgart, Germany
Antonio Bucchiarone, FBK-IRST of Trento, Italy
Jose Alfredo F. Costa, Universidade Federal do Rio Grande do Norte (UFRN), Brazil
Marc Kurz, Johannes Kepler University Linz - Institute for Pervasive Computing, Austria

ADAPTIVE 2015 Industry/Research Chairs

Dalimír Orfánus, ABB Corporate Research Center, Norway
Weirong Jiang, Xilinx Research Labs, San Jose, USA

ADAPTIVE 2015 Publicity Chairs

Kier Dugan, University of Southampton, UK

ADAPTIVE 2015

Committee

ADAPTIVE Advisory Chairs

Radu Calinescu, University of York, UK
Thomas H. Morris, Mississippi State University, USA
Serge Kernbach, CYBERTRONICA RESEARCH-Stuttgart, Germany
Antonio Bucchiarone, FBK-IRST of Trento, Italy
Jose Alfredo F. Costa, Universidade Federal do Rio Grande do Norte (UFRN), Brazil
Marc Kurz, Johannes Kepler University Linz - Institute for Pervasive Computing, Austria

ADAPTIVE Industry/Research Chairs

Dalimír Orfánus, ABB Corporate Research Center, Norway
Weirong Jiang, Xilinx Research Labs, San Jose, USA

ADAPTIVE Publicity Chairs

Kier Dugan, University of Southampton, UK

ADAPTIVE 2015 Technical Program Committee

Sherif Abdelwahed, Mississippi State University, USA
Nadia Abchiche-Mimouni, Université d'Evry, France
Habtamu Abie, Norwegian Computing Center/Norsk Regnesentral-Blindern, Norway
Muhammad Tanvir Afzal, Mohammad Ali Jinnah University- Islamabad, Pakistan
Jose M. Alcaraz Calero, University of the West of the Scotland, UK
Giner Alor Hernández, Instituto Tecnológico de Orizaba - Veracruz, México
Richard Anthony, University of Greenwich, UK
Flavien Balbo, Université Paris-Dauphine, Lamsade-CNRS, France
Bernhard Bauer, University of Augsburg, Germany
Christophe Bobda, University of Arkansas, USA
Jean Botev, University of Luxembourg, Luxembourg
Jesus G. Boticario, Spanish National University for Distance Education (UNED), Spain
Antonio Brogi, University of Pisa, Italy
Sven Brueckner, Axon, USA
Aldo Campi, Center for Industrial Research on ICT (CIRI ICT) - University of Bologna., Italy
Valérie Camps, IRIT-Toulouse, France
Radu Calinescu, University of York, UK

M. Alojzy Klopotek, Institute of Computer Science - Polish Academy of Sciences, Poland
Mitch Kokar, Northeastern University - Boston, USA
Satoshi Kurihara, Osaka University, Japan
Marc Kurz, Institute for Pervasive Computing, Johannes Kepler University of Linz, Austria
Rico Kusber, University of Kassel, Germany
Mikel Larrea, University of the Basque Country UPV/EHU, Spain
Ricardo Lent, Imperial College London, UK
Jingpeng Li, University of Stirling, UK
Henrique Lopes Cardoso, LIACC, Universidade do Porto, Portugal
Emiliano Lorini, Institut de Recherche en Informatique de Toulouse (IRIT), France
Sam Malek, George Mason University, USA
Célia Martinie, ICS-IRIT - University Toulouse 3, France
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal
Mieke Massink, FM&&T Group - CNR, Italy
Olga Melekhova, Université Pierre et Marie Curie - Paris 6, France
Frederic Migeon, IRIT/Toulouse University, France
Vivian Motti, Clemson University, USA
Gero Müehl, University of Rostock, Germany
Christian Müller-Schloer, Leibniz University of Hanover, Germany
Masayuki Murata, Osaka University, Japan
Filippo Neri, University of Naples "Federico II", Italy
Dirk Niebuhr, Clausthal University of Technology, Germany
Andrea Omicini, Università di Bologna, Italy
Flavio Oquendo, European University of Brittany/IRISA-UBS, France
Mathias Pacher, Leibniz Universität Hannover, Germany
Alexandros Paramythis, Contexity AG, Switzerland
Georg Püschel, Technische Universität Dresden, Germany
Raja Humza Qadir, dSPACE GmbH, Paderborn, Germany
Claudia Raibulet, University of Milano-Bicocca, Italy
Mahesh (Michael) S. Raisinghani, TWU School of Management, USA
Sitalakshmi Ramakrishnan, Monash University, Australia
Andreas Rausch, Technische Universität Clausthal, Germany
Wolfgang Reif, University of Augsburg, Germany
Brian M. Sadler, Army Research Laboratory, USA
Yacine Sam, Université François Rabelais Tours, France
Huseyin Seker, De Montfort University Leicester, UK
Sebastian Senge, TU Dortmund, Germany
Estefanía Serral, Vienna University of Technology, Austria
Marjan Sirjani, Reykjavik University, Iceland
Vasco Soares, Instituto de Telecomunicações / Polytechnic Institute of Castelo Branco, Portugal
Christoph Sondermann-Wölke, Universität Paderborn, Germany
Panagiotis Spapis, National and Kapodistrian University of Athens, Greece
Stephan Stilkerich, Airbus Group Innovations, Germany
Greg Sullivan, BAE Systems, USA

Javid Teheri, The University of Sydney, Australia
Christof Teuscher, Portland State University, USA
Luca Tesei, University of Camerino, Italy
Sotirios Terzis, University of Strathclyde, UK
Christof Teuscher, Portland State University, USA
Peppo Valetto, Drexel University, USA
Arlette van Wissen, VU University Amsterdam, Netherlands
Mirko Viroli, Università di Bologna, Italy

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

Structural Adaptations for Self-Organizing Multi-Agent Systems

Thomas Preisler and Wolfgang Renz

Multimedia Systems Laboratory
Faculty of Engineering and Computer Science
Hamburg University of Applied Sciences
Email: {thomas.preisler,wolfgang.renz}@haw-hamburg.de

***Abstract*—Over one decade of research in engineering of self-organization (SO) has established SO as the decentralized way to build self-adaptive systems. However, such SO systems even when well engineered may, under certain conditions, exhibit unwanted dynamical behavior, e.g. performance may decrease and/or starvation may occur. A promising concept to overcome such dynamical in-efficiencies in SO systems is to realize the dynamic exchange or reconfiguration of the coordination processes responsible for the self-organizing behavior in terms of a *structural adaptation*. In this paper, we propose an architecture and engineering approach to support the self-adaptive, structural exchange (or reconfiguration) of self-organizing coordination processes based on distributed Multi-Agent technology. Here, a sensor in each agent detects any decrease of specified SO performance indicators which initiates a distributed consensus process that allows for the exchange (or reconfiguration) of the self-organizing coordination processes, enabling the system to adapt to changing conditions automatically.**

Keywords–Self-Organizing Systems; Multi-Agent Systems; Structural Adaptation; Decentralized Coordination

I. INTRODUCTION

For self-organizing Multi-Agent Systems (MAS), the capability to adapt to a variety of (mostly external) influences, i.e., their *adaptivity*, is a key feature. In this context, adaptivity describes the ability of a system to change its structure respective behavior in response to external influences or altering demands. In addition, adaptive, self-organizing systems still strive towards reaching (initially defined) global goals. Looking in more detail into such adaptive systems, they reveal many different facets. According to [1] it can be distinguished between *design-time* and *run-time* adaptivity, where the latter one is far more challenging. Figure 1 depicts even more dimensions of adaptive systems. It differentiates between approaches that only change system parameters in order to exhibit adaptive behavior and concepts that can even alter the whole structure respective replace certain components of the system. Furthermore, it discerns between approaches based on centralized or decentralized architectures and on how the adaptivity is managed. Thereby, it is differentiated between solutions where the adaptivity is managed manually or automatically by the system itself. The red dot shown in the Figure ranks the proposed solution according to the different dimensions of adaptive systems. The approach is based on a decentralized architecture and emphasizes structure-based changes (while also supporting parameter-based changes). Possible adaptations are modeled manually at *design-time* and executes automatically at *run-time*. Therefore, the solution is ranked between manual and automatic adaptations.

Developing and operating systems that belong to this class of adaptive systems, i.e., self-organizing systems, is a challenging task. Firstly, it requires a systematic development approach that copes at all stages with three inherent characteristics of these systems: non-linear dynamics, stochastic behavior and emergent phenomena. Secondly, it requires a modular system architecture which enables the adaptation of the structure at runtime using highly customizable and reusable coordination processes. These coordination processes can be understood as standalone design elements that equip a self-organizing system with a specific dynamic behavior. By exchanging these coordination processes at runtime, a distributed application can adapt not only its behavior but also its inherent structure. Thus, enabling the system to overcome problems like performance decrease or starvation, by adjusting the structure of its self-organizing processes automatically. The approach presented in this paper extends already established approaches for engineering self-organizing systems by introducing a system architecture that systematically enables structural adaptations for distributed systems with decentralized control. It is comparable to the reactive planning approach (local) from the BDI (belief, desire, intention) agent architecture [2]. The system as a whole strives towards a distributed consensus (global) to execute predefined structural adaptations plans.

One of the many example of coordination processes in SO systems are distributed consensus algorithms. These algorithms establish self-organization and adaptivity, e.g., in multi-vehicle routing and also exhibit areas in parameter space where they get ineffective and need to get exchanged [3]. In contrast, we will use consensus methods on a meta-control level in this paper in order to detect thresholds for exchanging SO mechanisms (see Figure 2 in Section III).

Accordingly, the remainder of this paper is structured as follows: Section 2 describes related work, followed by Section 3 where the properties and engineering challenges of self-organizing MAS are described and a test scenario is presented. Section 4 introduces the core concept of structural adaptations before Section 5 concludes the paper.

II. RELATED WORK

Current trends in computer science like mobile and ubiquitous computing in combination with an increasing diversification of hard- and software platforms challenge traditional approaches of engineering and operating distributed systems substantially. Years ago, distributed systems were mainly closed systems with a-priori known tasks, challenges, and users. Nowadays, with an increasing pervasiveness of distributed systems, they have turned into an integral part of

 ISBN: 978-1-61208-391-9

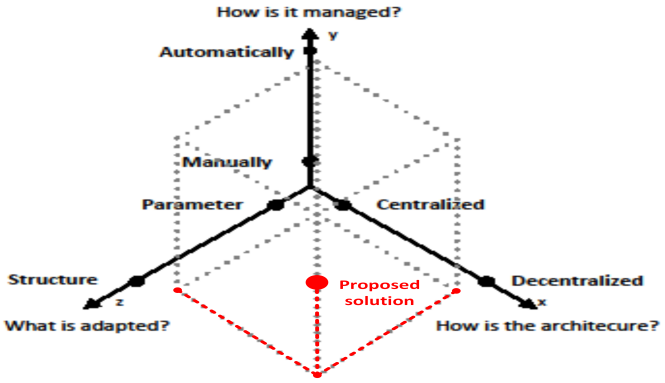


Figure 1. Dimensions of adaptive systems according to [4].

the business world as well as the private life of many people. This evolution implicates new challenges for distributed systems. For instance, they are forced to deal with high and unpredictable dynamics, an increasing complexity, and the satisfaction of non-functional requirements like, e.g., robustness, availability, and scalability. Altogether, this requires a new generation of distributed systems that is capable of *adapting* its behavior *autonomously*. This challenge is addressed by research areas like Autonomic [5] or Organic Computing [6] that aim at providing new approaches to solve it in a systematic fashion. They achieve it with different types of feedback loops, relying on (usually) centralized control elements. The authors of [7] identify feedback loops as the key design element within a distributed system in order to be able to exhibit adaptivity. Here, feedback loops consist of three main components: *sensor*, *actuators* and a *computing entity*. Sensors are in charge of observing the behavior and the (current) status of the system respective the environment it is situated in. Actuators can change the configuration of the system, which can either lead to *parametric* or *structural* changes. The computing entity which serves as a connector between the system input (sensor) and the output (actuator) can be very different w.r.t its internal architecture and abilities [8]. For instance, the widespread autonomic control loop [5], which is based on a monitoring, analyzing, planning, executing and knowledge loop (MAPE-K), contains a centralized computing entity, which can be associated with an autonomic manager to software and hardware components in order to equip them with adaptive behavior. Contrasting to feedback loops, [9] introduces a policy-based approach where the non-functional concerns of an application are described as policies and the application adapts itself to changing conditions controlled by a centralized policy engine. Like the approach presented in this paper [9] also focuses on a clean separation between the business-logic and the self-adapting fulfillment of non-functional requirements.

According to [10], this class of approaches, that introduce centralized control concepts, can be called *self-adaptive systems*. In contrast to this, there are approaches that aim at providing *automatically adaptive systems* which rely on decentralized architectures and utilize distributed feedback loops and coordination mechanisms. They are called *self-organizing systems* [10] and seem, due to their decentralized system architecture, to be better suited to deal with the aforementioned non-functional requirements. Also the concept of self-organization has been observed in many other domains like, e.g., biology, physics, or sociology and has, furthermore, proven its applicability for distributed systems already before (as, e.g., mentioned in [11] [12]).

In addition to the difference between centralized and decentralized feedback loops, another criterion targets the general applicability of existing approaches that aim at providing methods for structural adaptivity for distributed systems. According to [10], adaptivity defines a general system view that can be further decomposed into so called self-* properties of distributed systems. Therefore, there are many approaches which target the provision of a subset of these properties [13] [14] [15]. Whereas these approaches reach good results with respect to specific aspects of adaptive behavior, they lack general methods and concepts that provide structural adaptivity in general. This, however, limits the general applicability of these approaches and forces system developers to deal with (completely) different approaches if there is the requirement for more than just one type of adaptivity. Consequently, if a system requires different self-* properties it has to incorporate different concepts and techniques which increases the complexity of related implementations considerably.

This could be improved by using approaches which are based on structural adaptivity. However, applying them to the concept of self-organization in decentralized systems is an ambitious task. Especially the purposeful engineering of self-organizing systems is challenged by their inherent non-linear dynamics and the bottom-up development process. There is a lack of approaches (and corresponding implementations) that deal systematically with the whole development process. In contrast, approaches like [16] [17] focus mainly on early development activities as, e.g., requirement analysis or modeling, whereas approaches like [18] [19] provide basic implementation frameworks. Approaches like [20] [21] do provide a comprehensive development process but focus on self-adaptive systems based on centralized control concepts. An approach towards meta-adaptation support based on reusable and composable adaptation components is presented in [22]. The introduced *Transformer* framework constructs system global adaptation by contextually fusing adaptation plans from multiple adaptation components. Similar to the work presented in this paper, it focuses on decentralized structural adaptation for multiple purposes. While the work presented here focuses on a general engineering approach, the work presented in [22] focuses more on the conflict resolution between different adaptation behaviors.

In conclusion, it can be stated that there is a lack of approaches that combine the above mentioned aspects in order to support structural adaptivity as a basis for systematic development of generic self-organizing systems. Therefore, the following Section introduces an approach based on the systematical engineering of self-organizing MAS which supports the whole development process. It uses decentralized feedback structures and aims at supporting structural adaptivity in general.

III. SELF-ORGANIZING MAS

The presented approach on structural adaptation of coordinations processes is based on previous work on self-organizing dynamics in MAS. Such a self-organizing dynamic

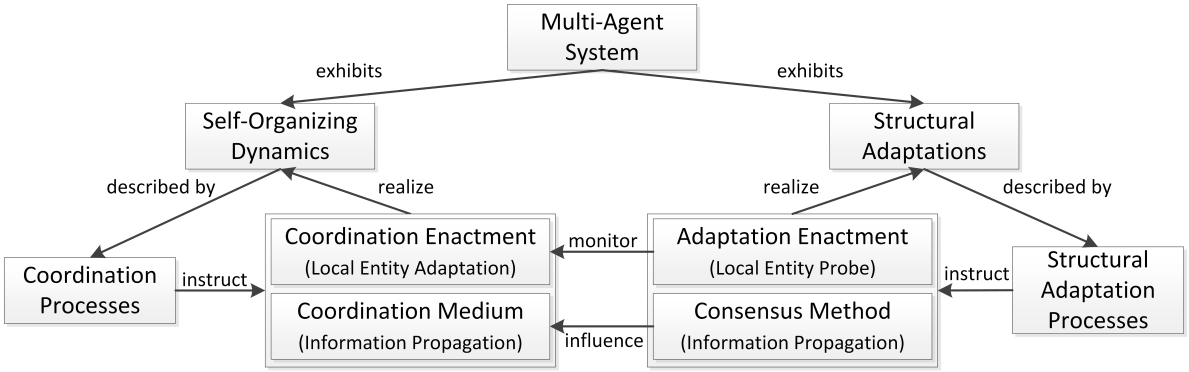


Figure 2. Structural adaptation processes exchange or reconfigure the coordination processes responsible for the self-organizing dynamics by monitoring SO performance indicators.

is shown in the left side of Figure 2. The MAS exhibits a self-organizing dynamic that causes the system to adapt to external and internal influences. The self-organizing dynamic realizes the intended adaptivity of the software system and is mapped by decentralized coordination processes. The processes describe a self-organizing behavior that continuously structures, adapts and regulates aspects of the application. They instruct sets of decentralized coordination media and coordination enactments. Coordination enactments and media distinguish between techniques for the adaptation of system elements (local entity adaptation) and realizations of agent interactions (information propagation). Together they control the microscopic activities of the agents, which on a macroscopic level lead to the manifestation of the intended self-organizing dynamic. The integration of the coordination enactments and media is prescribed by the coordination process definitions, which structure and instruct their operations. Thus the self-organizing dynamic of the MAS is described by coordination processes, which model the intended adaptivity of the system. On a technical level these processes instruct coordination enactments and media which realize the intended adaptivity.

Conceptually speaking, structural adaptations for self-organizing systems means an adaptation of the coordination processes, as they describe the system's intended self-organizing dynamic. The right side of Figure 2 illustrates this concept. The MAS exhibits structural adaptations which influence and observe the self-organizing dynamic. Similar to the self-organizing dynamic, the structural adaptations are described by processes. They define adaptation conditions, which specify states of the system where the self-organizing dynamic should be altered. This alteration is realized by prescribed adaptation activities, which are triggered due to specified SO performance indicators. Ineffective coordination processes are deactivated if the system's behavior becomes deficient and therefore other coordination processes are activated in exchange. This results in a structural adaptation of the coordination process composition. This means, on a lower level, if the system's behavior is not deficient but measured inefficient, the active coordination processes are reconfigured by parametric adaptations. The structural adaptation processes instruct adaptation enactments to monitor the agents with regard to the SO performance indicators. In both cases (structural or parametric adaptation), it is necessary that the system's entities find a consensus whether or not the adaptations should be performed. Therefore, a distributed consensus method is utilized in order to come to a decision about the execution of the adaptation. In case of a positive decision it is performed by manipulating the relevant coordination processes.

A. *Coordination Enactment Architecture*

The work on structural adaptations is based on a previously published tailored programming model for the software-technical utilization of coordination processes as reusable design elements [23]. The programming model provides a systematic modeling and configuration language called *MASDynamics* and a reference architecture to enable the enactment of pre-described coordination models called *DeCoMAS* [24] (Decentralized Coordination for Multi-Agent Systems). The architecture is based on the clean separation of activities that are relevant to the coordination of agents and the system's functionality. Therefore, coordination processes can be treated as first class design elements that define application-independent coordination interdependencies. Figure 3 illustrates the layered structure of the coordination enactment architecture. The functionality of the MAS is mapped by the application layer. The coordination logic is realized as an underlying layer. This layer provides a set of coordination media which provide the required coordination mechanisms. They build the infrastructure that allows the agents to exchange application independent coordination information and control the information dissemination. Thus, the coordination media are the technical realization of previous described coordination processes. The agents communicate with the coordination media using their coordination enactments (cf. Figure 3(B)). The enactments influence and observe the agent activities (1) and exchange information that is relevant for the coordination via the coordination media (2). The local configuration of these activities, e.g., when to publish information and how to process perceptions, is defined in a declarative, external coordination model (3) written in the *MASDynamics* language. Coordination is declaratively described to support the reuse of coordination pattern in different applications. This architecture focuses on the transparent separation of application and coordination logic, meaning that agent models are not modified by the coordination logic. This allows the supplement of coordination to existing applications. A recent example on how this architecture can be implemented to realize decentralized coordination in self-organizing systems based on Peer-to-Peer

technology is described in [25].

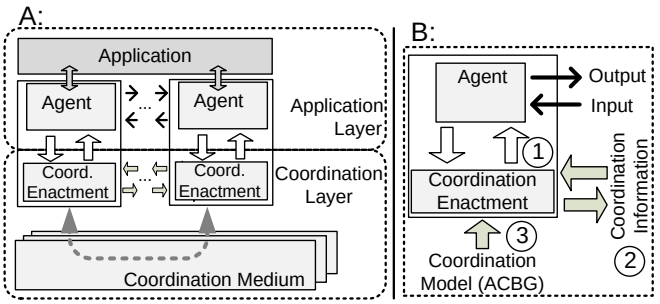


Figure 3. Coordination enactment architecture [26].

B. Engineering Self-Organizing MAS

The engineering of structural adaptations of coordination processes is part of an existing engineering approach for self-organizing MAS developed in the *SodekoVS* project [27]. Part of the project was the development of the *Coordination Enactment Architecture*. The project aims at providing self-organizing processes as reusable elements that developers can systematically integrate into their applications. The utilization of self-organization in software engineering is addressed by providing a reference architecture to offer a conceptual framework for the configuration and integration of self-organizing processes. The integration is guided by adjusting methodical development procedures. Following this, coordination media are made available as middleware services. A minimal intrusive programming model allows developers to configure and integrate representations of nature-inspired coordination strategies into their applications. Figure 4 denotes the conceptual view on integrating self-organization. Incremental development activities are supplemented with activities that address the manifestation of self-organizing phenomena (I-V). While developers design the functionality of their applications, they revise the decentralized coordination of component activities in interleaved development activities. Supplements to the requirements activities (I) facilitate the description of the intended application dynamics. During analysis activities (II) it is examined which instances or combinations of coordination metaphors can lead to the required adaptivity. Design activities (III) detail the models of selected coordination strategies and configure the coordination media that are used for their realization. These activities prepare the implementation and integration (IV) of medium instances to be configured and accessed by a generic usage interface. Testing (V) activities are supplemented with a simulation-based validation that agent coaction meets the given requirements, i.e., manifests the intended adaptiveness. The whole development process, as described in [27], is designed as a iterative process. Based on the results of the simulation-based validation, the self-organized coordination processes are redesigned until they achieve the intended adaptivity. Either based on the validation results or as an result of the initial analysis it may be observed that certain coordination processes or certain process configuration are only suitable for certain conditions but become deficient or insufficient for others. In this case it is practical to utilize structural adaptations for the re-composition of coordination processes or, on a lower level, parametric adaptations for the reconfiguration of coordination

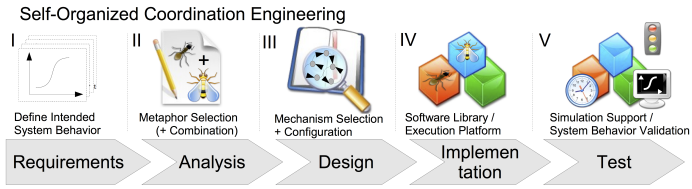


Figure 4. SodekoVS development activities following [27].

parameters. The key challenge hereby is to identify these conditions that require a structural (or parametric) adaptation and map them to SO performance indicators. As an addition to the existing engineering approach this paper propagates anticipated structural adaptations. Following the same iterative approach as designing and implementing the coordination processes, the adaptations should also be designed and implemented in a iterative way. The conditions that require adaptations should be identified based on the requirements and analysis activities and redefined by the results of a simulation-based validation.

C. Example: MarsWorld Coordination

This paper utilizes the MarsWorld scenario to explain the following concepts and their usage. The scenario is based on an application setting presented in [28]. A set of autonomous robots is sent to the planet of Mars to mine ore. The mining process consists of three distinct activities: (1) *analyze* locations to verify the presence of ore, (2) the mining or *production* of the ore and (3) *transporting* the mined ore to a homebase. The robots in this scenario are controlled by software agents, which are specialized to perform one of the distinct mining operations. The *Sentry* agents are equipped with sophisticated sensors to analyze potential ore locations, *Producer* agents have the capability to mine ore deposits at analyzed locations and the *Carry* agents can transport the mined ore to the homebase. Obviously as none of the three agents types is able to mine alone, the agents need to work together to achieve their collaborative goal. Therefore, this scenario is chosen for the case study, as the agents require some sort of coordination in order to achieve the collaborative goal.

The *Sentry* agents analyze potential ore deposits and inform the *Producer* agents whether or not ore can be mined there. The *Producers* mine the ore at the analyzed deposits and inform the *Carry* agents about it, so they can carry it to the homebase. Initially all agents explore the environment randomly. All agents have sensors to find potential ore deposits. If *Producer* or *Carry* agents encounter any potential deposits, they inform a *Sentry* agent. *Sentry* agents that have encountered a potential deposit or were informed about one, analyze the deposit. When they have verified the presence of ore at the location they request a *Producer* agent to mine the ore. Accordingly, after mining the ore *Producers* request a *Carry* agent to transport it to the homebase.

As the MarsWorld example exhibits no predefined organizational structure the agents need to be coordinated in order to achieve their collaborative goal. Based on the communication between the agents three coordination processes are needed to handle the information distribution:

1) Coordination information the *Producer* and *Carry* agents send to the *Sentry* agent, when they en-

countered potential ore deposits while exploring the environment.

2) Coordination information the *Sentry* agents send to *Producer* agents when they have analyzed a potential ore deposit and found ore to mine there.
3) Coordination information the *Producer* agents send to the *Carry* agents after ore was mined and is now ready for transportation to the home base.

As a simple example a coordination process manifestation based on a neighborhood approach is envisioned. In this case, each coordination process selects the agent of the appropriate type, which is nearest to the emitting agent. By selecting the nearest agent it is ensured that the informed agents have to travel the minimal distance to reach the designated destination. If the environment consists of multiple ore deposit clusters, characterized by a small distance between the ore deposits in the cluster and a large distance to the next cluster, this coordination approach lets the agents form local groups in a self-organized way. Thus, the different agent types will organize themselves in an emergent way.

Of course the coordination processes based on this approach require knowledge about the current positions of each agent. In the case study, the environment is equipped with a positioning service offering this information. The three coordination process realizations utilize the service in order to distribute the coordination messages to the nearest matching agent. This results in the realizations of the following three coordination processes:

- *latest_target_seen_nearest*: Whenever a *Producer* or *Carry* agent has encountered a potential ore deposit, this coordination process selects the *Sentry* agent nearest to the location of the *Producer* or *Carry* agent and informs it about the deposit. The *Sentry* agent adds the location to its queue and eventually analyzes it.
- *latest_target_analyzed_nearest*: After a *Sentry* has analyzed a potential ore deposit and actually found ore there, this coordination process selects the *Producer* agent nearest to the location and calls it to mine ore here.
- *latest_target_produced_nearest*: When a *Producer* agent has completely depleted the ore deposit, this coordination process selects and informs the *Carry* agent nearest to the location, so that it can transport the mined ore to the homebase.

Technical speaking, the coordination processes are described declaratively with the *MASDynamics* language. The *DeCoMAS* framework initializes the *Coordination Enactments* for the participating agents at application start time. These enactments monitor the agents and whenever one of the described *Coordination Events* occurs, the according *Coordination Information* is sent to the nearest matching agent. This is done by facilitating a *Coordination Medium*, which makes use of the environments positioning service to determine the nearest matching agent. The *Coordination Enactment* of the receiving agent triggers a *Coordination Event*, when the *Coordination Information* is received, letting the agent react based on its defined behavior.

IV. Structural Adaptation of Coordination Processes

This Section presents the structural adaptation architecture and the extension of the *MASDynamics* language for describing the adaptations in a declarative way. Also, an example for a structural adaptation based on the MarsWorld scenario is introduced.

A. Adaptation Architecture

The previous described Coordination Enactment architecture was extended by the introduction of so called *Adaptation Enactments*, to enable structural adaptations of coordination processes. Similar to Coordination Enactments, which were introduced to equip applications with coordination capabilities, the Adaptation Enactments equip applications with the capability to structural adapt coordination processes at runtime. Figure 5 shows the extension of the Coordination Enactment architecture. The Adaptation Enactments are part of the coordination layer and therefore independent from the system's functionality (supporting a clean separation between application and coordination logic). They observe the agents similar to the Coordination Enactments. But contrasting to the Coordination Enactments they do not influence the agents, but the coordination media, which are the technical realization of the coordination processes. As described before, the Coordination Enactments influence and observe the agent activities and exchange information relevant for coordination via a coordination medium. As shown in the Figure, the Adaptation Enactments consists of two components. The *Monitor* observes specified SO performance indicators as part of the agent, and in case of a decreased performance, determined by a specified condition, it initiates a consensus process for structural adaptations. The *Service* is used as an interface for the distributed consensus process. It offers generic consensus interface methods to support different consensus approaches (e.g., voting).

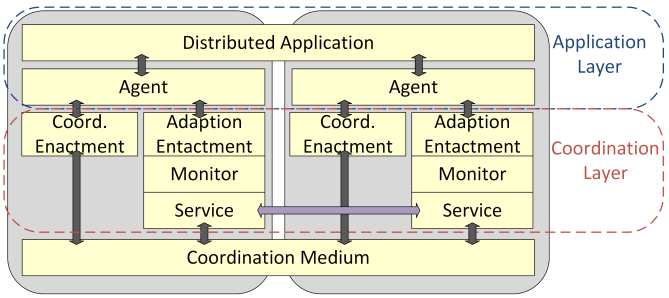


Figure 5. Adaptation architecture.

An example for a simple consensus algorithm is the following voting scheme. When the Adaptation Enactment Monitor of an agent observes a decreased performance, it initializes a distributed voting attempt and acts as leader of this vote. Utilizing the service interface it presents a suggested adaption to the other agents. Based on their local information (state of the agent), the called Adaptation Enactment Services decide whether or not to agree on the proposed adaptation and inform the vote leader about their decision. The leader analyzes the received votes and decides if the required majority for the

adaptation was reached. If so the suggested adaption is committed by activating or deactivating the affected coordination media, respective by changing their coordination parameters.

B. Adaptation Description

To describe structural adaptations the *MASDynamics* language was extended to support the declarative description of possible adaptations. These adaptations are described at design time and executed at runtime. As described before, structural adaptation of coordination processes can be realized by exchange or reconfiguration of coordination processes. Therefore, the already realized description of coordination processes was extended to indicate whether or not a coordination process is active at start time. This allows a developer to define multiple coordination processes with different behavior at design time, from whom only a subset may be active at start time. Furthermore, the *MASDynamics* language was extended with the following elements to describe these possible adaptations: The first part concerns the types of agents that are allowed to participate in the adaptation process. The *DeCoMAS* framework creates Adaptation Enactments for this types of agents, so they can be monitored with regards to specified performance indicators and are able to participate in the decision making process.

The second part concerns the actual adaptations. Each adaptation is identified by a unique *id* and a *reset* flag. The reset flags specifies if the performance indicator should be reset after a failed adaptation attempt. It can be used to prevent repeated adaptation attempts flooding the system, if only a subset of agents are subjected to bad performance indicators, while the majority of the systems still performs well and in that case would not agree on an adaptation. Further information in the adaptation description concerns the consensus process. It includes a minimum total number of *answers* a vote leader awaits, before it starts to evaluate the results from an adaptation attempt it started. Also the required *quorum* that has to be reached, before a structural adaptation may be accepted is specified. A *timeout* after which the vote leader will start to evaluate it, even if it has not received the required number of answers can also be specified. Furthermore, it is possible to *delay* an adaptation attempt if specified. Also, an adaptation can be blocked for a certain amount of time at start time, to avoid oscillation problems (*startDelay*).

Besides the information concerning the adaptation process, the description also contains information about the affected coordination processes. This includes the *realization id* of the affected coordination processes and the information if the process should be *activated* or *deactivated*, respective which *parameter* of the coordination process should be altered to which *value*. The last information needed for structural adaptations are the *constraints* regarding the agents state. For each agent type, they specify which *element* should be used as a SO performance indicator. They contain a *condition* and a *threshold*. The condition is used by the Adaptation Enactments Monitor, to point out if the performance indicator has become deficient for the specified agent and therefore, if an adaptation attempt should be started. When the Adaptation Enactment Service of an agent receives an adaptation request, it uses the threshold to determine whether or not it should agree to the proposed adaptation. Therefore, the threshold allows the specification of an insufficient performance indicator that is not as as strict as the actual condition, which would force the agent to start an adaptation attempt by itself. The threshold maps a negative trend allowing the agent to anticipate an insufficient performance.

C. Example: MarsWorld Structural Adaptation

In order to test the structural adaptations at runtime, the MarsWorld scenario was extended with an other manifestation of the three coordination processes described in Section III-C. The new manifestation is based on a simple random selection approach and therefore, does not exhibit any self-organizing behavior. In this case, each of the coordination processes randomly selects an agent of the appropriate type and informs it about the sensed, analyzed or produced ore.

Arguably, the coordination process manifestations that are based on the neighborhood approach will perform better, as the formation of local mining groups around the ore clusters minimizes the distance the agents have to travel, before they can analyze, produce or transport ore and therefore, optimizes the overall mining efficiency. But these coordination processes depend on the positioning service offered by the environment. If this service fails, the coordination processes will not be able to select the nearest agents, thus, they will not be able to inform the according agents. In such a case the application would benefit from the capability to structurally self-adapt and to switch to the random-selection based coordination process manifestations.

The goal is to deactivate the location based coordination processes, when the positioning service offered by the environment fails. As compensation the random selection based coordination processes will be activated. The agents have no knowledge about the fact that the positioning service has failed in this scenario conception. But they can measure the time that has passed since they have received the last coordination information. If they have not received any messages within a given time, they assume that something went wrong and the positioning service is broken. In this example, an agent waits 20 seconds until it assumes a malfunction and initializes a voting process acting as leader. Of course it is possible that agents have not received any coordination information within this time frame for other reasons. Therefore, they have to find a consensus, whether or not the coordination processes should be adjusted to overcome local phenomena. Agents that have received a voting requests determine if they have received any coordination information within the last 15 seconds (threshold) and if so vote *yes*. The agent, which started the voting process waits until it has received all the voting results (this is a simplification because in this small scenario we neglect any message lost) and evaluates them. If the required majority of 75% has been reached, the structural adaptation is performed. At start time, the adaptation capability is blocked for 30 seconds, because it may take a while before the first potential ore deposits are sensed and therefore, the system might adapt prematurely because of oscillation problems.

To measure the impact of the structural self-adaptation 50 simulations run with the location-based coordination processes active at start time were executed. After 60 seconds a failure in the environment's positioning service was simulated. Thus, the coordination processes were no longer able to select the nearest agent and therefore, were not able to distribute the coordination information. The adaptation capability was blocked for 30

 ISBN: 978-1-61208-391-9

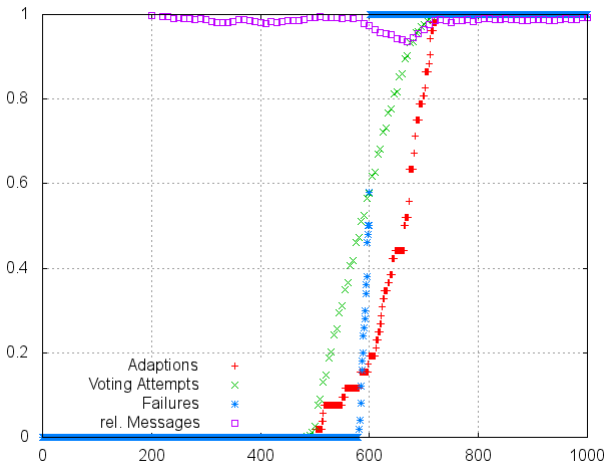


Figure 6. Evaluation of the Structural Adaptations for the MarsWorld (axis are described in the text).

seconds at start time, so 50 seconds had to pass before first voting attempts were started. Figure 6 shows the evaluation results of the self-adaptation. The x-axis denotes the time in 1/10 second steps. The y-axis shows the total number of voting attempts, the percentage of simulated failures in the positioning service and the percentage of adaptations. The Figure shows how the simulated failure in the location service occurred after 60 seconds have passed (deviations are caused by inaccuracies of the MAS platform's clock). It took 50 seconds before first voting attempts occurred. This is the shortest possible amount of time been passed before agents could observe the absence of any coordination information within the last 20 seconds. As Figure 6 shows, the percentage of voting attempts grows until all systems have performed the adaptation. In this case the adaptation was configured as *unique*, so after it was performed, no further voting attempts were undone.

The results also show that in approximately 50% of the simulation runs voting attempts occurred before the actual failure has arisen. This behavior can be explained by the random exploration of the environment and the communication cascades. Carry agents only receive coordination information after sentry agents have sensed ore and producer agents have mined it. If only a few ore deposits have been sensed after 50 seconds, the majority of the agents will not have received any coordination information until this time and therefore, interpret the absence of such messages as a potential failure. Thus, voting for an adaptation. A higher blocking time for the adaptation capability at start time or higher observation and threshold values would lead to fewer premature adaptations. On the other hand this would lead to a higher response time before the system adapts itself after a failure. As described before, suitable parameters have to be identified as part of the iterative, simulation driven engineering approach. The Figure also shows the relative number of coordination messages in relation to the number of coordination messages from a scenario were no failure occurs and therefore, no adaptation was needed. The curved line shows that there is no significant deviation between the two scenarios until the failure occurred. When the failure occurs the number of messages slides down in relation to the failure-free scenario. After the self-adaptation took place, the number of messages rises again to the level of the failure-free scenario. This shows how the self-adaption is able to repair a deficient behavior.

V. Conclusion and Future Work

In this paper, we presented an architecture and engineering approach for structural adaptations in self-organizing MAS to realize the dynamic exchange or reconfiguration of self-organizing coordination processes. It aims at supporting structural adaptations in general, rather than focusing on single self-* properties. The approach consists of a generic system architecture that governs the development of self-organizing MAS and a description language that supports the declarative description of coordination processes and pre-described structural adaptations, which can be processed by the corresponding framework automatically. It is supported by an engineering process consisting of incremental development activities that are supplemented with activities that address the manifestation of self-organizing behavior. The approach supports the modularization of coordination, which enables reusability and interoperability of coordination processes. It propagates a clear separation between application functionality and coordination, allowing developers to implement coordination without the need to change the applications business logic. Furthermore, with the introduction of structural adaptations of coordination processes it supports the self-adaptive structural exchange or reconfiguration of self-organizing processes. By detecting any decrease of specified SO performance indicators, the Adaptation Enactment extension initiates a distributed consensus process, that allows for the exchange or reconfiguration of the self-organizing processes to adapt to changing conditions automatically. Since the approach uses a set of coordination processes to be defined at design time, it is conceptually comparable to the reactive planning approach in a single BDI agent (local planing), but it strives towards finding a global structural adaptation plan for the system as a whole.

The presented framework was used in the *MarsWorld* scenario to realize a collaborative application in which three different types of agents needed to be coordinated, in order to mine ore on Mars. Therefore, three different coordination processes with two different manifestations (random or proximity-based) were implemented. The scenario was used as a proof of concept, to show how structural adaptations of coordination processes can be used. A specified SO performance indicator pointed out that an agent has not received any coordination messages within a given time, which led to an exchange of coordination processes at runtime.

Arguably, this work is still in progress and therefore, both the implemented distributed voting scheme as a consensus method and the MarsWorld scenario are kept quite simple and offer room for improvement. Future work will focus on these two aspects. A more sophisticated distributed consensus algorithm with a strong focus on decentralized coordination will be developed to replace the proof of concept voting scheme. Also the approach will be tested on more realistic and complex scenarios (e.g., the self-organized redistribution of bicycles in a bike sharing system [29]) involving different types of structural adaptations, in order to analyze which scenarios will profit from such adaptations and to improve the overall engineering approach.

 ISBN: 978-1-61208-391-9

ACKNOWLEDGMENT

The authors would like to thank Deutsche Forschungsgemeinschaft (DFG) for supporting this work through a research project on "Self-organization based on decentralized coordination in distributed systems" (SodekoVS).

REFERENCES

[1] K. Geihs, "Selbst-adaptive software," Informatik-Spektrum, vol. 31, 2008, pp. 133–145.

[2] A. S. Rao and M. P. George, "Bdi agents: From theory to practice," in Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95), 1995, pp. 312–319.

[3] W. Ren and R. W. Beard, Distributed consensus in multi-vehicle cooperative control : theory and applications, ser. Communications and control engineering. London: Springer, 2008.

[4] A. Vilenica, "Anwendungsentwicklung selbstorganisierender systeme: Systematische konstruktion und evaluierung," Ph.D. dissertation, Hamburg University, Department of Informatics, Vogt-Kölln-Str. 30, 22527 Hamburg, Germany, 12 2013.

[5] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no. 1, 2003, pp. 41–50.

[6] J. Branke, M. Mnif, C. Müller-Schloer, H. Prothmann, U. Richter, F. Rochner, and H. Schmeck, "Organic computing - addressing complexity by controlled self-organization," in Proc. of the 2th Int. Symp. on Leveraging Appl. of Formal Methods, Verification and Validation, ser. ISOLA '06. IEEE Comp. Soc., 2006, pp. 185–191.

[7] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw, "Software engineering for self-adaptive systems through feedback loops," B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, ch. Engineering Self-Adaptive Systems through Feedback Loops, pp. 48–70.

[8] J.-P. Mano, C. Bourjot, G. A. Lopardo, and P. Glize, "Bio-inspired mechanisms for artificial self-organised systems," Informatica (Slovenia), vol. 30, no. 1, 2006, pp. 55–62.

[9] L. Veiga and P. Ferreira, "Poliper: policies for mobile and pervasive environments," in Proceedings of the 3rd workshop on Adaptive and reflective middleware, ser. ARM '04. New York, NY, USA: ACM, 2004, pp. 238–243.

[10] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Trans. Auton. Adapt. Syst., vol. 4, May 2009, pp. 14:1–14:42.

[11] M. Mamei, R. Menezes, R. Tolksdorf, and F. Zambonelli, "Case studies for self-organization in computer science," J. Syst. Archit., vol. 52, no. 8, 2006, pp. 443–460.

[12] T. D. Wolf and T. Holvoet, "A catalogue of decentralised coordination mechanisms for designing self-organising emergent applications," Dept. of Comp. Science, K.U. Leuven, Tech. Rep. CW 458, 8 2006.

[13] D. Garlan, "Model-based adaptation for self-healing systems," in In Proceedings of the first workshop on Self-healing systems. ACM Press, 2002, pp. 27–32.

[14] M. Smit and E. Stroulia, "Autonomic configuration adaptation based on simulation-generated state-transition models," in Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conf. on, 2011, pp. 175 –179.

[15] E. Yuan and S. Malek, "A taxonomy and survey of self-protecting software systems," in Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on, june 2012, pp. 109 –118.

[16] M. Morandini, F. Migeon, M.-P. Gleizes, C. Maurel, L. Penserini, and A. Perini, "A goal-oriented approach for modelling self-organising mas," in Proceedings of the 10th International Workshop on Engineering Societies in the Agents World X, ser. ESAW '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 33–48.

[17] T. De Wolf and T. Holvoet, "Towards a methodology for engineering self-organising emergent systems," in Proc. of the 2005 conference on Self-Organization and Autonomic Informatics (I). Amsterdam, The Netherlands: IOS Press, 2005, pp. 18–34.

[18] G. Di Marzo Serugendo, J. Fitzgerald, and A. Romanovsky, "Metaself: an architecture and a development method for dependable self-* systems," in Proc. of the 2010 ACM Symp. on Applied Comp., ser. SAC '10. New York, NY: ACM, 2010, pp. 457–461.

[19] S.-W. Cheng, "Rainbow: cost-effective software architecture-based self-adaptation," Ph.D. dissertation, School of Computer Science, Carnegie Mellon Universit, Pittsburgh, PA, USA, May 2008.

[20] K. Geihs, P. Barone, F. Eliassen, J. Floch, R. Fricke, E. Gjorven, S. Hallenstein, G. Horn, M. Khan, A. Mamelli, G. Papadopoulos, N. Paspallis, R. Reichle, and E. Stav, "A comprehensive solution for application-level adaptation," Software: Practice and Experience, 2008.

[21] S. Hallsteinsen, K. Geihs, N. Paspallis, F. Eliassen, G. Horn, J. Lorenzo, A. Mamelli, and G. Papadopoulos, "A development framework and methodology for self-adapting applications in ubiquitous computing environments," Journal of Systems and Software, vol. 85, no. 12, Dec. 2012, pp. 2840–2859.

[22] N. Gui and V. De Florio, "Towards meta-adaptation support with reusable and composable adaptation components," in Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on, Sept 2012, pp. 49–58.

[23] J. Sudeikat and W. Renz, "MASDynamics: Toward systemic modeling of decentralized agent coordination," in Komm. in Vert.Syst. (KiVS). Springer, 2009, pp. 79–90.

[24] ——, "Decomas: An architecture for supplementing mas with systemic models of decentralized agent coordination," in IEEE/WIC/ACM International Conference on Web Intelligence. IEEE, 2009, pp. 104 – 107.

[25] T. Preisler, A. Vilenica, and W. Renz, "Decentralized coordination in self-organizing systems based on peer-to-peer coordination spaces," in Works. on Self-organising, adaptive, and context-sensitive distributed systems (SACS), vol. 56, 2013.

[26] A. Vilenica, J. Sudeikat, W. Lamersdorf, W. Renz, L. Braubach, and A. Pokahr, "Coordination in Multi-Agent Systems: A Declarative Approach using Coordination Spaces," in Proc. of the 20th European Meeting on Cybernetics and Systems Research (EMCSR 2010) - Int. Workshop From Agent Theory to Agent Impl. (AT2AI-7), R. Trappl, Ed. Austrian Soc. for Cyb. Studies, 4 2010, pp. 441–446.

[27] J. Sudeikat, L. Braubach, A. Pokahr, W. Renz, and W. Lamersdorf, "Systematically Engineering Self-Organizing Systems : The SodekoVS Approach," Electronic Communications of the EASST, vol. 17, 2009, p. 12.

[28] J. Ferber, Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence, 1st ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[29] T. Preisler, W. Renz, and A. Vilenica, "Bike-sharing system reliability problems - a data-based analysis and simulation architecture," in Sustainability and Collaboration in Supply Chain Management: A Comprehensive Insight into Current Management Approaches, ser. Supply Chain, Logistics and Operations Management, C. M. R. Wolfgang Kersten, Thorsten Blecker, Ed., vol. 16. Josef Eul Verlag, 8 2013, pp. 83–97.

 ISBN: 978-1-61208-391-9

On the Impact of Routers' Controlled Mobility in Self-Deployable Networks

Karen Miranda
Universidad Autónoma Metropolitana (UAM)
Dept. of Applied Mathematics and Systems
Cuajimalpa, Mexico City
Email: kmiranda@correo.cua.uam.mx

Nathalie Mitton and Tahiry Razafindralambo
Inria Lille - Nord Europe
Villeneuve d'Ascq, France
Email: {Name.Surname}@inria.fr

***Abstract*— A substitution network is a temporary network that self-deploys to dynamically replace a portion of a damaged infrastructure by means of a fleet of mobile routers. In this paper, we evaluate the performance of a previous self-deployment scheme, adaptive positioning algorithm (APOLO), for substitution networks and we show the benefit of the controlled mobility in such a network. To that end, we evaluate APOLO in terms of throughput under several scenarios and different metrics. These results constitute a comprehensive evaluation of APOLO and enable to envision new ways of optimization and future paths of research. We prove that APOLO is an efficient deployment and redeployment algorithm for mobile relay networks.**

Keywords–Substitution Networks; Robot deployment; Controlled mobility.

I. INTRODUCTION

A Rapidly Deployable Network (RDN) is a solution to provide communication services in disaster scenarios. Specifically, we focus on a wireless solution named the substitution networks [1]. A substitution network is a temporary network to replace a portion of a damaged infrastructure (called hereafter base network) by means of mobile routers (called substitution routers) capable of moving on demand and connecting to the base network through bridge routers (Figure 1a).

Bridge routers are connected in between the base and the substitution networks, and used to forward the traffic from the base network to the substitution network and vice versa.

Mobile substitution routers are wireless routers of the substitution network, possibly connected to bridge routers, and whose union provides alternative path(s) to the base network.

Figure 1 depicts the complete overview of using a Substitution Network, where the bridge routers are deployed together with the base network (Figure 1a). In this example, the base network operates without the help of the mobile routers. When a failure occurs (Figure 1b), the mobile routers are deployed. In this architecture, the failure detection and the deployment are done autonomously by the base network itself. Mobile routers try to find an optimal position to restore the connectivity service and to ensure Quality of Service (QoS) (Figure 1c). In some cases, the continuous redeployment of the routers may be necessary to adapt to an evolving network and to QoS conditions (see Figure 1d).

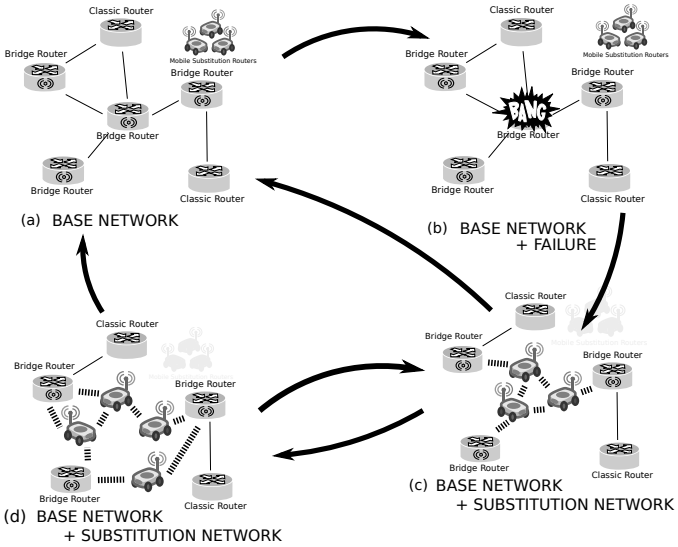


Fig. 1. Typical use case for a base network and a substitution network.

Particularly, our goal is to have an autonomous router deployment, as well as a possible redeployment of the mobile routing devices. Therefore, it is necessary to design algorithms and protocols to deploy and re-deploy such devices [2]. Since the routing devices are autonomously provided with a limited battery, it is also necessary to consider energy constraints during the deployment. Moreover, the deployment computation process does not consider a central entity in the network, hence, this process should be executed in a distributed manner [3]. An efficient router/relay (mobile or static) deployment algorithm must take the link quality into account in order to decide when and where to deploy a relay [4]. For that purpose, the deployment algorithms must be able to measure the wireless link quality. We have presented in a previous work the Adaptive POsitioning aLgOrithm (APOLO) [5].

This paper presents the extended results of APOLO [5] developed for substitution networks obtained under stress and some results under different assumptions, especially regarding channel states. The remaining of the paper is structured as follows. After browsing the literature in Section II, Section III describes the background and the basic concepts used in this paper. Then, Section IV depicts the proposed algorithm APOLO. Section V and Section VI present the simulation settings and results. Finally, we discuss these results and conclude in Section VII.

 ISBN: 978-1-61208-391-9

II. State of the art

A self-deployable network is composed of several mobile routers. Such mobile routers perform the same tasks as their static similars distributing the data traffic; however, the mobile routers must self-position in a given area. We can find in the literature few proposals to self-deploy a network. In [2], the authors present the Spreadable Connected Autonomic Network (SCAN) algorithm where the mobile routers move to expand the covered area. In order to maintain connectivity, mobile routers are allowed to move as long as there is no risk of disconnection, if a possible disconnection is detected, the mobile routers must stop moving. A different strategy is presented in [3], where the mobile routers follow a leader router on a straight-line formation, once the leader reaches the desired point it stops and the rest of the routers start stopping as well. Finally, Kim et al. [6] propose a string-type formation to deploy the mobile routers, all with identical characteristics. Each mobile router seeks and adjusts its position according to the packet delivery ratio or bandwidth. These approaches differ from ours in the sense that they do not consider link quality. Such proposals have succeeded to autonomously deploy a network; nevertheless, an efficient self-deploy algorithm should adapt dynamically to the environment changes, for example, different zone sizes, changes on mobile users' distribution, or changes on the channel conditions.

III. Preliminaries

We consider a wireless network composed of mobile routers that are located and may move on the two-dimensional Euclidean space. We use "node" as a generic term for any device in the simulation neighborhood, for instance, the mobile or classic routers. For the sake of simplicity, we assume that the transmission range R of a node u is the area in which another node v can receive/send messages from/to u, i.e., $d(u,v) < R(u)$, where $d(u,v)$ represents the Euclidean distance between u and v, and therefore, it exists a link X between u and v. We assume that two nodes are "neighbors" when they are within the communication range of each other.

In the following, we use X_{prev} and X_{next} to refer to the links of the previous and next hops, respectively, of a mobile router. Likewise, we assume that some of the devices are fixed, that traffic needs to be transferred between two fixed devices, and that the wireless routers dynamically move in the scenario and act as relays, regardless of the routing protocol. And, as many link layer protocols, we assume that each node is equipped with a timer and an 802.11 wireless card, as well as with an identifier that is unique in the network (MAC address).

We define the quality of a communication link, or just "link quality", as the probability that a message transmitted on the link is successfully received, that is, the reliability of the link [4]. The link quality can be assessed as a function of the received signal strength (RSS) or the signal-to-noise ratio (SNR) [7], for example. In general, higher SNR leads to lower probability of error in the packet. Hence, a link with high SNR is considered a high-quality link [8]. We use the RSS, SNR, RTT, and TxRate as values to measure the link quality because their values retrieve insight of the performance of a wireless network [9]. Therefore, we call RSS, SNR, RTT, and TxRate "link metrics" or "link parameters" in general.

We use the term "broadcast" to refer to the message propagation in a router's neighborhood in order to obtain the link measurements. Also, we refer to the control packets of routing protocols as "hello" messages or beacons and to the packets used in active measurements as probe packets. Finally, we define the term *controlled mobility* as the ability of some nodes to move by themselves to a specific destination or with a specific goal, i.e., the opposite of randomly [10].

In this paper, we use the Dynamic Source Routing (DSR) [11] protocol for our set of simulations. The DSR protocol is a self-maintaining routing protocol designed for multi-hop wireless networks composed of mobile nodes. DSR uses on demand routing allowing each source to determine the route used to transmit its packets to the corresponding destinations.

IV. Adaptive Positioning Algorithm

A new quality-of-service-based architecture for substitution networks, presented in [1], envisions a wireless network composed of mobile routers/relays that provide alternative paths to the base network. Such substitution routers are able to move on demand, so, they can self-deploy and adapt to the network topology accordingly to the environment conditions.

Previously, we have presented APOLO for self-deploying mobile routers in a substitution network [5]. During the substitution network lifetime, APOLO is executed in each mobile router to determine whether it has to move by using the feedback of the link quality coming from one-hop neighbors. APOLO consists of three major stages. Firstly, APOLO measures the link quality by means of one link parameter, e.g., RSSI, SNR, or delay. Secondly, APOLO computes the gathered data and makes the movement decision, i.e., if the router needs to move or not to improve the link quality. And finally, APOLO determines direction of the movement and the router moves accordingly.

The results presented in [5][12] are restricted to only one propagation model (two-ray ground) and three simulation scenarios. In this paper, we extend these results with extra propagation models and scenarios. First, we present the results obtained when the router's initial position is random. Second, we study the behavior of our algorithm when there exist multiple mobile routers between the source and the destination. Then, we evaluate the redeployment capacity of the mobile router when a new node arrives. Finally, we compare the deployment performance of each link parameter under three different propagation models (two-ray ground, shadowing, and ricean).

V. Simulation environment

We present an extension to the experimental performance evaluation of APOLO. Our main goal is to present the effectiveness of the algorithm to deploy wireless mobile routers in a given area. To that end, we evaluate our proposal by using the

 ISBN: 978-1-61208-391-9

TABLE I. SIMULATION PARAMETERS.

Physical	Propagation	Two Ray Ground
	Error Model	Real [15]
	Antennas Gain	$G_t = G_r = 1$
	Antennas Height	$h_t = h_r = 1$ m
	Min Received Power	$P_{r-thresh}$=6.3 nW
	Mobile Router Energy	50 J
	Communication Range	240 m
MAC	802.11b	Standard Compliant
	Basic Rate	2Mbps
	Auto Rate Fallback	1, 2, 5.5, 11 Mbps
LLC	Queue size	50 pkts
	Policy	Drop Tail
Routing	Static	Dijkstra [16]
	Routing Traffic	None
Transport and Application	Flow	CBR / UDP
	Packet Size	512 B/1 MB
Statistics	Number of samples	$k = 10$
	Broadcast period	$t = U(0.1)$
Mobility	Movement step	$d = 2m$

NS-2 network simulator [13]. In our previous work [12], we have chosen three scenarios proposed in [14] as a result of the study on relay wireless networks. We propose three additional scenarios plus a propagation model comparison.

Since our goal is to assess the impact of controlled mobility in wireless routers, we use the DSR protocol for our simulations, although, APOLO is not tied to any routing particular protocol. Table I summarizes the basic parameters used in our simulations in NS-2. In this paper, we use the instantaneous throughput (TH_{ins}) defined as he number of bits transferred to the final destination in any given instant to assess the performance of our deployment algorithm.

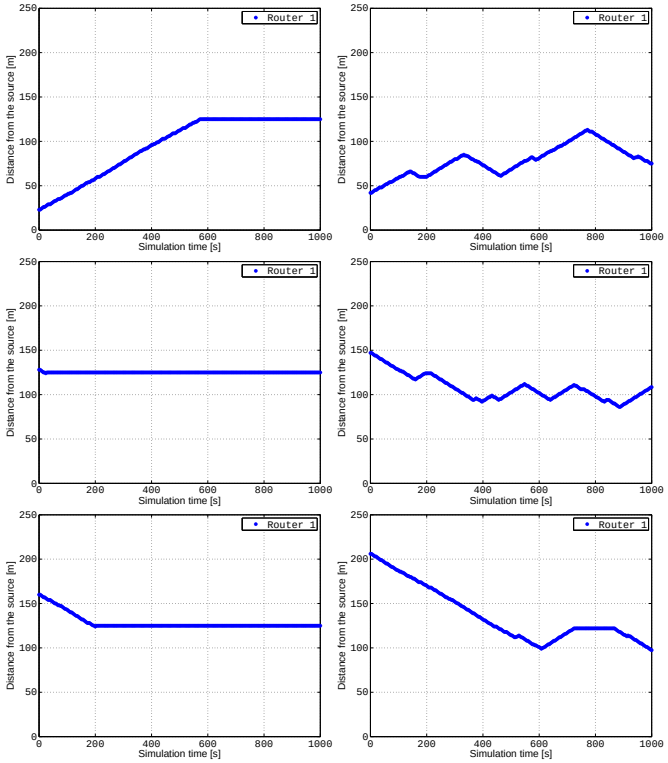
VI. RESULTS

A. Random initial position

In our previous work, the basic scenario is composed of one mobile router, one source node (S), and a destination node (D). The source and the destination are placed 250 m away from each other, i.e., the source is placed at coordinates (0,0) and the destination at coordinates (250,0). Finally, at the beginning of the simulation, the router node is placed 10 m away from the source node, that is, at coordinates (10,0), as depicted in Figure 2. Then, the router starts moving based on APOLO. Nevertheless, the mobile router may reach the position that equalizes the parameter values regardless of its initial position.

Hence, we take again this basic scenario. We run a set of simulations choosing the initial position of the router randomly. We use the 802.11b standard along with the two-ray ground model. Figure 3 plots the x coordinates along time to illustrate the movement evolution. Figure 3(a) presents the results while using the RSS as link metric and Figure 3(b) presents the results while using the SNR as link metric. In Figure 3(a) the router's initial coordinates are (23,0), (128,0), and (160,0), respectively. The theoretical position that equalizes should be (125,0) where the router is placed at exactly

Fig. 2. Basic scenario composed of one mobile router, one source node, and one destination node.



(a) Results by using the RSS as link metric (b) Results by using the SNR as link metric

Fig. 3. Simulation results for random initial position of the mobile router.

the middle point between the source and the destination, and therefore, the link values of X_{prev} and X_{next} should be the same. We observe in Figure 3(a) that the router reaches such a position regardless of its initial position, an expected result since the two-ray ground model calculates the RSS as a function of the distance. Similarly, in Figure 3(b), the router's initial coordinates are (42,0), (147,0), and (206,0), respectively. Despite the router does not reach a steady position, this behavior corresponds to the previous simulations where the router's initial position is (10,0), in other words, the initial position of the router does not affect the movement behavior by using SNR as link metric. We believe that such a behavior is due to the propagation model we have used. In order to corroborate this, we perform a set of simulations with different propagation models. The results are presented in Section VI-E.

B. Multiple routers scenario

In [12], we studied the performance of APOLO by considering a scenario with two routers. In such a scenario, there is a source and a destination communicating though two mobile routers, so, we extend this scenario by considering three and four intermediate routers between the source and the destination, as shown in Figure 4.

1) Three routers: Regarding the three router scenario, the source, once again, is placed at coordinates (0,0) and the destination at coordinates (300,0), that is, 300 m away from each other. At the beginning of the simulation, Router 1 is placed at coordinates (10,0), Router 2 is placed at (150,0),

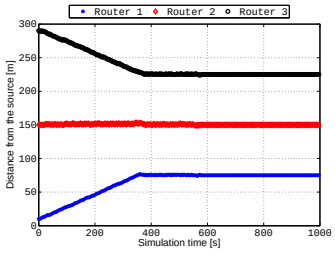
and Router 3 is placed at (290,0). Then, the routers execute APOLO to adjust their position using the RSS as link metric, the results are presented in Figure 5. The deployment evolution is plotted as x coordinates as function of simulation time where we observe that the behavior of the routers follows the results obtained with the two router scenario (Figure 5(a)). The routers travel the corresponding distance to equalize the values of the link metric. In this case, Router 1 and Router 3 travel a similar distance. The routers reach their final position after 550 s, which are, Router 1 at 75 m, Router 2 at 150 m, and Router 3 at 225 m from the source, that means a distance of 75 m between each node. Figure 5(b) plots the instantaneous throughput obtained during the routers deployment.

2) Four routers: Regarding the four router scenario, the source is placed at coordinates (0,0) and the destination at coordinates (400,0), that means, 400 m away from each other. At the beginning of the simulation, Router 1 is placed at coordinates (10,0), Router 2 is placed at (140,0), Router 3 is placed at (260,0), and Router 4 is placed at (390,0). Subsequently, each router adjusts its own position by executing APOLO using RSS as link metric. Figure 6(a) shows the routers deployment through the simulation time. Once again, the final position of the routers equalize the link metric and it is equidistant between nodes. These results, three and four routers, show that the behavior experienced with only one mobile router is duplicated, and therefore, APOLO is a scalable solution and it is able to deal with multiple router scenarios.

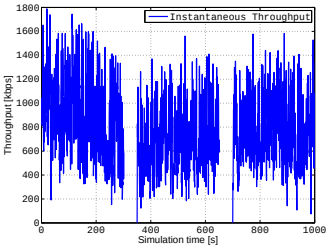
C. Two sources and one destination scenario

By the same token, we evaluate APOLO in a multiple destination scenario [5]. In addition, we present a topology with multiple sources by using the topology depicted in Figure 7, where we illustrate two sources (`n0`, `n1`) and one destination (`n3`) out of range. Thus, a mobile router (`n2`) is used to connect the sources and the destination. At the beginning of the simulation, the router (`n2`) is placed 10 m away from the source nodes (`n0`, `n1`) on the straight line

Fig. 4. Multiple routers scenario composed of three mobile routers, one source node, and one destination node.

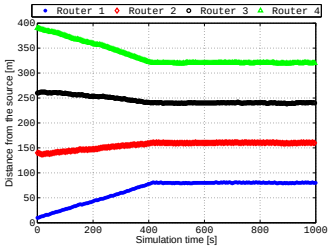


(a) Movement evolution in time of the mobile routers

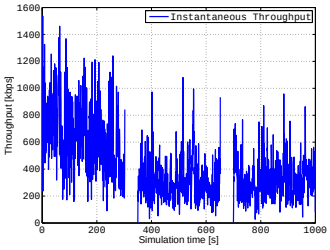


(b) Throughput as a function of time

Fig. 5. One source, one destinations, and three mobile routers.



(a) Movement evolution of the routers



(b) Throughput as function of the time

Fig. 6. Multiple routers scenario composed of four mobile routers, one source node, and one destination node.

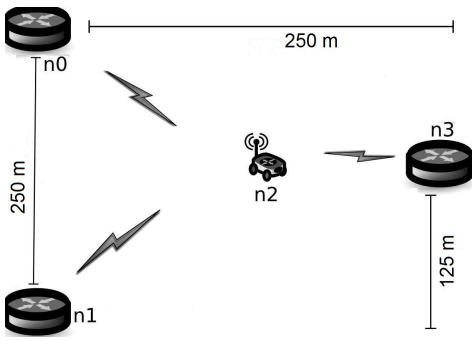


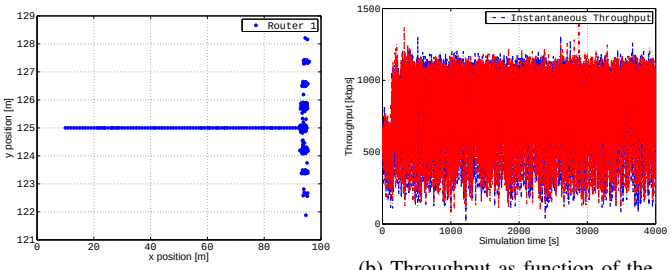
Fig. 7. Two sources, one destination, and one mobile router scenario.

that connects the sources from the middle position between the receiver node (`n3`), and finally, we use User Datagram Protocol packets with a size of 1 MB since UDP is used in most multimedia applications.

In this scenario, two identical CBR/UDP flows are transmitted from the source nodes (`n0`, `n1`) to destination node (`n0`) starting and finishing at the same time. A priori, we assume that the best position is located at the coordinates (83,125), which is the triangle centroid. Then, the router uses APOLO to decide where to move and RSS as link metric. The movement trace is depicted in Figure 8(a). During the first 500 s, the router moves in a straight line from (10,125) to (94,125), i.e., it travels 84 m. After this time, the router moves from left to right (in the y axis domain) in a range of ± 3 m, i.e., from (94,128) to (94,122). This behavior is very different from that one experimented in a similar scenario with two source nodes and one destination. In the latter scenario, the router moves mostly in the x axis range and stops very close to the triangle centroid. Hence, the oscillation experienced in two source scenario is caused by the two flows arriving to the router, that means, the router moves to improve the link quality accordingly to the data flow arriving at the time. In order to avoid wasting energy with short distance movements, it is possible to implement a movement threshold, for example, the router moves if the RSS value drops under certain threshold.

D. Redeployment

One issue that remains open is the routers redeployment. The redeployment is specially important in dynamic scenarios. In the following, we present a simulation campaign where the redeployment is needed. As in the previous set of simulations,



(a) Movement evolution of the router

(b) Throughput as function of the time

Fig. 8. Scenario two sources and one destination. Movement of the mobile router through the time and the corresponding throughput.

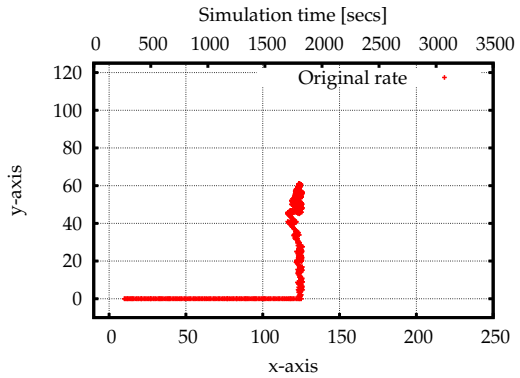


Fig. 9. Redeployment of a mobile router.

the mobile router executes APOLO to redeploy and uses RSS as link metric. At the beginning of the simulation, the scenario is the one depicted in Figure 2, one source node, one destination node, and one mobile router. The source (`n0`), the router (`n1`), and the destination (`n2`) are located at coordinates (0,0), (10,0), and (250,0), respectively. Then, after 650 s a new source node (`n3`) arrives to transmit data to (`n2`), and therefore, the router must adapt its position. Node (`n3`) appears at coordinates (0,125).

Figure 9 plots the movement of the router in the Cartesian space. The first seconds of the simulation, the router follows the same behavior than in the previous simulation, in other words, the router reaches the position at (125,0). Then, when the second source appears, the router starts to move to equalize the quality of the new link between (`n3`) and (`n1`). Finally, the router reaches its final position at coordinates (124,60). These results are very interesting for two reasons. Firstly, they prove that APOLO is useful to redeploy automatically mobile routers, and hence, it is well suited in dynamic scenarios. And secondly, the results prove the importance of the hello messages to advertise the eventual changes in the network topology, if the transmission rate of such hello messages is too low, the information gathered may not reflect the changes in the topology over the time. Nevertheless, it is also important to consider the cost of such hello messages in terms of energy and overhead. To overcome this problem, it is interesting to study the optimal transmission frequency of the hello packets [17], as well as the overhead reduction techniques.

E. Propagation model comparison

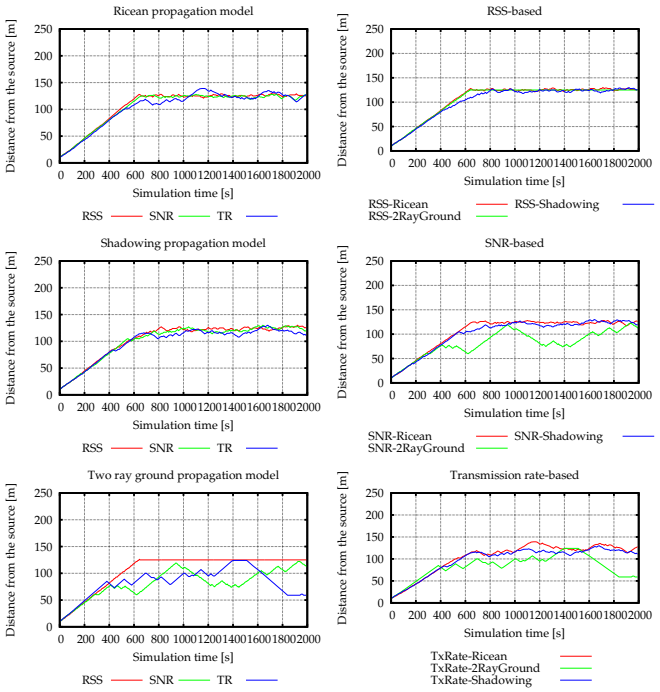
The propagation models are empirical mathematical formulations to characterize the radio wave propagation based on physical phenomena such as distance, used frequency, or fading effects. In wireless network simulations, the propagation models are used to simulate the wireless channel by computing the wireless signal strength at the receivers for any packet transmitted by a single sender. Particularly, NS-2 simulator provides three propagation models: Free space model, Two Ray Ground model, and Shadowing model. Moreover, it is possible to add a fourth model called Ricean.

Each propagation model computes the attenuation of the signal strength between the sender and the receiver by using a Carrier Sensing Threshold (CSThresh_). If the signal strength is lower than CSThresh_, the packet is discarded at the physical layer. Otherwise, the signal strength is compared to a second threshold at the receiver (RxThresh_) to determine whether the packet is received with errors or not. If so, the MAC layer discards the packet.

Free space model represents the transmission range as a perfect circle around the sender. Basically, if the receiver is within the circle, it will receive all the packets; otherwise, it loses them. Two-ray ground reflection model considers both the direct path and a ground reflection path, a difference from the free space model, which only considers a single line-of-sight path. The two-ray ground model gives more accurate prediction at a long distance than the free space model. Nevertheless, the free space model is used when the distance is small because the two-ray model does not give a good result for a short distance due to the oscillation caused by the constructive and destructive combination of the two rays. It is important to notice that both models represent the communication range as a prefect circle, i.e., the received power is a deterministic function of distance. On the other hand, the shadowing model takes the fading phenomenon into account. Finally, the Ricean model characterizes the effect of small-scale fading (Rayleigh and Ricean). Such a fading is caused by movement of the sender, receiver, or of other objects in the environment. This movement may be characterized by the Doppler spreading.

As in the previous sections, we evaluate the performance of APOLO under three different propagation models, Ricean, Two ray ground, and Shadowing. To that end, we use the one source-one destination scenario (Figure 2), the router is placed at coordinates (10,0). Then, we vary the propagation model for each of the link parameters, i.e., RSSI, SNR, RTT, and transmission rate. The results obtained are presented in Figure 10. Figure 10(a) presents the comparison of the router deployment by using all the APOLO variants for each model propagation. The Two-ray ground plot was presented in [5], where router using the RSS variant reaches exactly the middle point between the source and the destination, i.e., at coordinates (125,0) while the router using the SNR and TxRate does not reach a steady point. However, this behavior changes completely when we evaluate APOLO by

 ISBN: 978-1-61208-391-9



(a) Comparison of the link parameters for each propagation model.

(b) Comparison of the each parameter for all the propagation models.

Fig. 10. Evaluation of APOLO under different propagation models. The movement of the mobile router as function of time.

using the Ricean and Shadowing models. Under both models, Ricean, and Shadowing the performance of the three variants is similar. None of the variants reaches a steady point still the router moves ± 10 m away from the middle point (125,0). Nonetheless, this behavior confirms our observation about the SNR performance obtained in Section VI-A, that is, since each propagation model characterizes in different way the events at the physical layer, the values of the link metrics correspond to such characterization and behave accordingly. We also include a comparison of each link parameter under all the propagation models to clarify the difference in the behavior (Figure 10(b)). Because the link quality values depend largely on the propagation model, it is important to choose the one that characterizes better our case of study. In general, the NS-2 community uses the two ray ground model but the shadowing model corresponds better to the real scenarios.

VII. Conclusion and Future Work

We have presented the impact of controlled mobility to self-deploy routers in substitution networks. The results provide a wide view of the performance of the Adaptive POsitioning aLgOrithm (APOLO). They prove that APOLO is able to successfully deploy the mobile routers in several scenarios and by using different propagation models. Moreover, APOLO performs well also when router redeployment is needed. The comparison of different deployment algorithms is not a trivial task since each proposal considers different metrics such as coverage, number of nodes, devices connected, delay, or throughput. We believe that a careful election of the performance metrics is the next step to continue on this direction. We have already set the metrics and we are currently comparing several deployment algorithms by means of simulations. Eventually, we will evaluate such algorithms by means of a real implementation with WiFiBots®.

We observed that by using the Ricean and Shadowing models the deployment performance of the SNR and TxRate variants outperform the two-ray ground one even if in the former case the router does no reach a steady point. This disadvantage may be overcome by adding a threshold to avoid useless movements. Thus, we are interested in studying how the threshold choice may (or may not) impact the performance.

VIII. Acknowledgment

This work was partially supported by a grant from CPER CIA and the ANR RESCUE project.

References

[1] T. Razafindralambo and others, “Promoting Quality of Service in Substitution Networks with Controlled Mobility,” in *ADHOC-NOW*, Paderborn, Germany, 2011, pp. 248–261.

[2] J. Reich, V. Misra, D. Rubenstein, and G. Zussman, “Connectivity Maintenance in Mobile Wireless Networks via Constrained Mobility,” IEEE JSAC, vol. 30, no. 5, pp. 935–950, Jun. 2012, pp. 935–950.

[3] C. Q. Nguyen and others, “Using Mobile Robots to Establish Mobile Wireless Mesh Networks and Increase Network Throughput,” *IJDSN*, vol. 2012, 2012, pp. 1–13.

[4] M. R. Souryal, A. Wapf, and N. Moayeri, “Rapidly-Deployable Mesh Network Testbed,” in Proc. Globecom, Honolulu, USA, 2009, pp. 1–6.

[5] K. Miranda, E. Natalizio, and T. Razafindralambo, “Adaptive Deployment Scheme for Mobile Relays in Substitution Networks,” IJDSN, vol.2012, 2012, pp. 1–9.

[6] K.-H. Kim, K. G. Shin, and D. Niculescu, “Mobile Autonomous Router System for Dynamic (Re)formation of Wireless Relay Networks,” IEEE Transactions on Mobile Computing, vol. 12, no. 9, 2013, pp. 1828–1841.

[7] N. Baccour and others, “F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks,” in Proc. of EWSN, Coimbra, Portugal, 2010, pp. 240–255.

[8] S. Farahani, ZigBee Wireless Networks and Transceivers. Newton, MA, USA: Newnes, 2008.

[9] E. Feo Flushing, J. Nagi, and G. A. Di Caro, “A mobility-assisted protocol for supervised learning of link quality estimates in wireless networks,” in Proc. of ICNC, Hawaii, USA, 2012, pp. 137–143.

[10] E. Natalizio and V. Loscrì, “Controlled mobility in mobile sensor networks: advantages, issues and challenges,” Telecommunication Systems, vol. 52, no. 4, pp. 2411–2418, Apr. 2013, pp. 2411–2418.

[11] D. B. Johnson, D. A. Maltz, and J. Broch, “Ad Hoc Networking.” Addison-Wesley Longman Publishing Co., Inc., 2001, ch. DSR: The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, pp. 139–172.

[12] K. Miranda, E. Natalizio, T. Razafindralambo, and A. Molinaro, “Adaptive router deployment for multimedia services in mobile pervasive environments,” in Proc. WIP of PerCom, Lugano, Switzerland, 2012, pp. 471–474.

[13] NS, “Network Simulator v.2.29 (NS-2),” accessed on January 2015. [Online]. Available: http://isi.edu/nsnam/ns/

[14] L.-L. Xie and P. Kumar, “Multisource, Multidestination, Multirelay Wireless Networks,” IEEE Transactions on Information Theory, vol. 53, no. 10, Oct. 2007, pp. 3586–3595.

[15] J. del Prado Pavon and S. Chio, “Link adaptation strategy for IEEE 802.11 WLAN via received signal strength measurement,” in Proc. of ICC, Anchorage, USA, 2003, pp. 1108–1113.

[16] E. Dijkstra, “A note on two problems in connexion with graphs,” Numerische Mathematik, vol. 1, no. 1, 1959, pp. 269–271.

[17] X. Li, N. Mitton, and D. Simplot-Ryl, “Mobility Prediction Based Neighborhood Discovery in Mobile Ad Hoc Networks,” in Proc. of NETWORKING, Valencia, Spain, 2011, pp. 241–253.

 ISBN: 978-1-61208-391-9

Coordination-level Adaptation in Distributed Systems

Ichiro Satoh
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ichiro@nii.ac.jp

***Abstract*—This paper proposes a framework for adapting the behaviors and coordinations of software agents to changes in distributed systems. It is unique to other existing approaches for self-adaptation because it introduces the notions of differentiation in cellular into real distributed systems. When an agent delegates a function to another agent coordinating with it, if the former has the function, this function becomes less-developed and the latter's function becomes well-developed. The framework was constructed as a middleware system and allowed us to define agents as Java objects written in JavaBean. We present several evaluations of the framework in a distributed system instead of any simulation-based systems.**

***Keywords*-Differentiation; Distributed System; Coordination.**

I. INTRODUCTION

Distributed systems are dynamic and complicated by nature and may partially have malfunctioned, e.g., network partitioning. Their scale and complexity are beyond the ability of traditional management approaches, e.g., centralized and top-down ones. Distributed systems should adapt themselves to changes in their system structures, including network topology, and the requirements of their applications. Many researchers have explored adaptations for distributed systems. The author proposed an approach to adapting behaviors in software components that a distributed application consisted of without any centralized and top-down management systems [9]. The approach was inspired by a natural adaptation, called *cellular differentiation*, which was a mechanism by which cells in a multicellular organism become specialized to perform specific functions in a variety of tissues and organs. The approach could differentiate their functions according to their roles in whole applications and resource availability, as just like cells. It involves function matching as differentiation factors in functions, where service matching is the process of comparing the function request against the available function advertisements and determining which function best satisfies the request. When a component delegates a function to another component, if the former has the function, its function becomes less-developed in the sense that it has less computational resources, e.g., active threads, and the latter's function becomes well-developed in the sense that it has more computational resources.

This paper presents the second step of our approach. The previous approach supports adaptations only at internal behaviors of software components [9], because it could not adapt coordinations between components. The proposed approach presented in this paper aimed at adaptations in coordinations between more than one component in addition to the internal behaviors of components. Although existing attempts for adaptive coordinations between computers have been explored, our approach has two notable advantages. The first is to manage adaptive coordinations without any centralized and top-down management systems. The second is keep consistency between adaptations in multiple computers, because adaptive coordinations need to be supported at more than one computer. Adaptations for distributed systems have several unique requirements that adaptaions for non-distributed systems do not have. Since a distributed system consists of multiple computers, adaptations tend to affect behaviors on more than one computer. However, adaptations may not be synchronized. Therefore, when some computers achieved their adaptations, others may not yet. In particular, adaptations for coordinations between multiple computers needs their adaptations need to be synchronized.

The remainder of this paper is structured as follows. In Section II, we present the related work. Section III discusses the basic approach and Section IV presents the design and implementation of our proposal. Section V evaluates the implementation and Section VI describes applications. Section VII gives some concluding remarks.

II. RELATED WORK

This section discusses several related studies on software adaptation in distributed systems. Existing can be classified into four types, parameter-level, software-level, location-level, and coordination-level adaptations. Although the first and second approaches are common in non-distributed systems, the third and fourth are available only in distributed systems.

The first is to adapt system parameters, e.g., durations of timeouts and the amount of available resources to changes in distributed systems. Adaptations in the type tend to be limited. The second is to adapt software for defining components running on computers. It enables behaviors of software in distributed systems to be adapted to changes in the systems. As mentioned previously, our previous approach is in the type. One of the most typical approaches of the type

 ISBN: 978-1-61208-391-9

is genetic programming [6]. The fitness of every individual program in the population need to be evaluated in each generation and multiple individuals are stochastically selected from the current population based on their fitness. However, since distributed systems may have an effect on the real world and be used for mission-critical processing, there is no chance of ascertaining the fitness of randomly generated programs. Blair et al. [1] tried to introduce self-awareness and self-healing into a CORBA-compatible Object Request Broker (ORB). Their system was a meta-level architecture with the ability of dynamically binding CORBA objects. The aim of resource management strategy is to maximize the profits of both customer agents and resource agents in large datacenters by balancing demand and supply in the market. Several researchers have addressed resource allocation for clouds by using an auction mechanism. For example, Lin et al [7] proposed a mechanism based on a sealed-bid auction. The cloud service provider collected all the users' bids and determined the price. Zhang et al. [12] introduced the notion of spot markets and proposed market analysis to forecast the demand for each spot market.

The third is location-level adaptations. Suda et al. proposed bio-inspired middleware, called Bio-Networking, for disseminating network services in dynamic and large-scale networks where there were a large number of decentralized data and services [8], [10]. Although they introduced the notion of energy into distributed systems and enabled agents to be replicated at and moved to suitable computers according to the number of service requests from clients, where the selection of computers depends on distances between agents and clients. As most of their parameters, e.g., energy, tended to depend on a particular distributed system, so that they may not have been available in other systems. Our approach should be independent of the capabilities of distributed systems as much as possible.

The fourth is coordination-level adaptations. Jaeger et al. [4] introduced the notion of self-organization to ORB and a publish/subscribe system. Georgiadis et al. [3] presented connection-based architecture for self-organizing software components on a distributed system. Like other software component architectures, they intended to customize their systems by changing the connections between components instead of internal behaviors inside the components. Cheng at al. [2] presented an adaptive selection mechanism for servers by enabling selection policies, but they did not customize the servers themselves. They also needed to execute different servers simultaneously. There have been many attempts to apply self-organization into distributed systems, e.g., a myconet model for peer-to-peer network [11]. There has been no attempts in the third and fourth types that keep consistency between adaptions at multiple computers.

III. Basic approach

This paper introduces the notion of (de)differentiation into a distributed system as a mechanism for adapting coordinations between software components, which may be running on different computers connected through a network.

Differentiation-inspired coordination-level adaptation: This mechanism was inspired by differentiation in dictyostelium discoideum. When dictyostelium discoideum cells aggregate, they can be differentiated into two types: prespore cells and prestalk cells. Each cell tries to become a prespore cell and periodically secretes chemical substance, called cAMP, to other cells. If a cell can receive more than a specified amount of the substance from other cells, it can become a prespore cell. There are three rules. 1) the substance chemotaxically leads other cells to prestalk cells. 2) A cell that is becoming a prespore cell can secrete a large amount of the substance to other cells. 3) When a cell receives more substance from other cells, it can secrete less substance to other cells.

Each agent has one or more functions with weights, where each weight corresponds to the amount of the substance and indicates the superiority of its function. Each agent initially intends to progress all its functions and periodically multicasts *restraining* messages to other agents federated with it within the domain of current networks. Restraining messages lead other agents to degenerate their functions specified in the messages and to decrease the superiority of the functions. As a result, agents complement other agents in the sense that each agent can provide some functions to other agents and delegate other functions to other agents that can provide the functions. Finally, functions that are often delegated to other agents, and then become inactive in the sense that they lose their computational resources.

Consistency in distributed adaptations: Coordination-based adaptations often need to modify protocols and application-logics in multiple computers. Such modifications are often required to be synchronized. Suppose an adaptation for coordination between two computers. While the first computer achieved its modification for the adaptation and another does not yet, if coordination between them happen, their coordination may be inconsistent, because the protocol or application-logic of one computer does not match with that of the another. To solve this problem, we introduce a synchronization mechanism for blocking coordinations among computers until the computers that do the coordinations complete their adaptations.

IV. Design and Implementation

Our approach is maintained through two parts: runtime systems and agents. The former is a middleware system for running on computers and the latter is a self-contained and autonomous software entity. It has three protocols for (de)differentiation and delegation.

 ISBN: 978-1-61208-391-9

A. Adaptive agent

Each agent is an autonomous programmable entity and consists of one or more functions, called the *behavior* parts, and its state, called the *body* part, with information for (de)differentiation, called the *attribute* part. These parts are implemented as a set of Java objects. We can define each agent as a single JavaBean, where each method in JavaBean needs to access the database maintained in the body parts.

- The body part maintains program variables shared by its behaviors parts like instance variables in object orientation. When it receives a request message from an external system or other agents, it dispatches the message to the behavior part that can handle the message.
- The behavior part defines more than one application-specific behavior. It corresponds to a method in object orientation. As in behavior invocation, when a message is received from the body part, the behavior is executed and returns the result via the body part.
- The attribute part maintains descriptive information with regard to the agent, including its own identifier. The attributes contains a database for maintaining the weights of its own behaviors and for recording information on the behaviors that other agents can provide.

The agent has behaviors $b_1^k, \ldots, b_n^k$ and w_i^k is the weight of behavior b_i^k. Each agent (k-th) assigns its own maximum to the total of the weights of all its behaviors. The W_i^k is the maximum of the weight of behavior b_i^k in k-th agent. The maximum total of the weights of its behaviors in the k-th agent must be less than W^k. ($W^k \geq \sum_{i=1}^{n} w_i^k$), where $w_j^k - 1$ is 0 if w_j^k is 0. The W^k may depend on agents. In fact, W^k corresponds to the upper limit of the ability of each agent and may depend on the performance of the underlying system, including the processor. Note that we never expect that the latter will be complete, since agents periodically exchange their information with neighboring agents. Furthermore, when agents receive no retraining messages from others for longer than a specified duration, they remove information about them.

The approach offers two communication policies for intercomponent interactions.

- If a component declares a *forward* policy for another, when specified messages are sent to other components, the messages are forwarded to the latter as well as the former.
- If a component declares a *delegate* policy for another, when specified messages are sent to the former, the messages are forwarded to the latter but not to the former.

The former policy is useful when two components share the same information and the latter policy provides a master-slave relation between them. The framework provides three interactions: publish/subscribe for asynchronous event passing, remote method invocation, and stream-based communication as well as message *forward* and *delegate* policies.

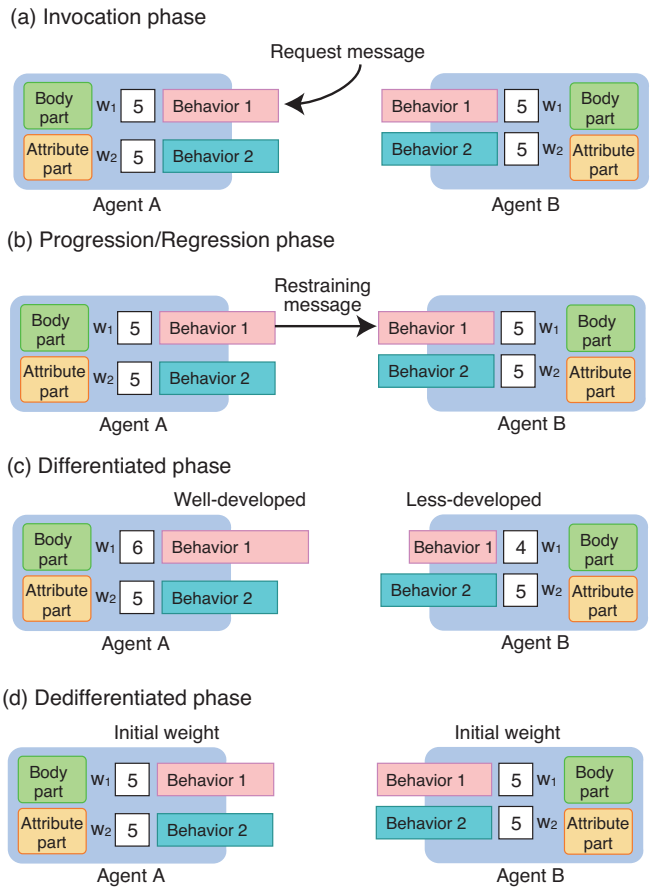


Figure 1. Differentiation mechanism for agent

B. Adaptive coordination

Next, we describe our differentiation-inspired adaptation mechanism.

1) Removing redundant functions: Behaviors in an agent, which are delegated from other agents more times, are well developed, whereas other behaviors, which are delegated from other agents fewer times, in a cell are less developed. Finally, the agent only provides the former behaviors and delegates the latter behaviors to other agents.

1: When an agent (k-th agent) receives a request message from another agent, it selects the behavior (b_i^k) that can handle the message from its behavior part and dispatches the message to the selected behavior (Figure 1 (a)).
2: It executes the behavior (b_i^k) and returns the result.
3: It increases the weight of the behavior, w_i^k.
4: It multicasts a restraining message with the signature of the behavior, its identifier (k), and the behavior's weight (w_i^k) to other agents (Figure 1 (b)).

The key idea behind this approach is to distinguish between internal and external requests. When behaviors are invoked

 ISBN: 978-1-61208-391-9

by their agents, their weights are not increased. If the total weights of the agent's behaviors, $\sum w_i^k$, is equal to their maximal total weight W^k, it decreases the minimal (and positive) weights (w_j^k is replaced by $w_j^k - 1$ where $w_j^k = \mathbf{min}(w_1^k, \ldots, w_n^k)$ and $w_j^k \geq 0$). The above phase corresponds to the degeneration of agents.

1: When an agent (k-th agent) receives a restraining message with regard to b_i^j from another agent (j-th) , it looks for the behaviors ($b_m^k, \ldots b_l^k$) that can satisfy the signature specified in the receiving message.
2: If it has such behaviors, it decreases their weights ($w_m^k, \ldots w_l^k$) and updates the weight (w_i^j) (Figure 1 (c)).
3: If the weights ($w_m^k, \ldots, w_l^k$) are under a specified value, e.g., 0, the behaviors ($b_m^k, \ldots b_l^k$) are inactivated.

C. Invocation of functions

When an agent wants to execute a behavior, even if it has the behavior, it needs to select one of the behaviors, which may be provided by itself or others, according to the values of their weights.

1: When an agent (k-th agent) wants to execute a behavior, b_i , it looks up the weight (w_i^k) of the same or compatible behavior and the weights ($w_i^j, \ldots, w_i^m$) of such behaviors ($b_i^j, \ldots, b_i^m$).
2: If multiple agents, including itself, can provide the wanted behavior, it selects one of the agents according to selection function ϕ^k, which maps from w_i^k and $w_i^j, \ldots, w_i^m$ to b_i^l, where l is k or $j, \ldots, m$.
3: It delegates the selected agent to execute the behavior and waits for the result from the agent.

The approach permits each agent to use its own evaluation function, ϕ, because the selection of behaviors often depends on its application and coordination. Although there is no universal selection function for mapping from behaviors' weights to at most one appropriate behavior like a variety of creatures, we can provide several functions.

D. Increasing resources for busy functions

The approach also provides a mechanism for duplicating agents, including their states, e.g., instance variables, as well as their program codes and deploying a clone at a runtime system. It permits each agent (k-th agent) to create a copy of itself when the total weights ($\sum_{i=1}^{n} w_i^k$) of functions ($b_1^k, \ldots, b_n^k$) provided in itself is the same or more than a specified value. The sum of the total weights of the original agent and those of the clone agent is equal to the total weights of the original agent before the agent is duplicated. The current implementation supports two conditions. The first permits each agent (k-th) to create a clone of it when the total of its weights ($\sum_{i=1}^{n} w_i^k$) is more than its maximal total weight W^k and the second condition is twice that of the total initial weights of the functions. When a busy agent running as a user program has no access resources, it allocates resources to the clone agent via the external control system.

E. Releasing resources for redundant functions

Each agent (j-th) periodically multicasts messages, called *heartbeat messages*, for a behavior (b_i^j), which is still activated with its identifier (j) via the runtime system. When an agent (k-th) does not receive any heartbeat messages with regard to a behavior (b_i^j) from another agent (j-th) for a specified time, it automatically decreases the weight (w_i^j) of the behavior (b_i^j), and resets the weight (w_i^k) of the behavior (b_i^k) to be the initial value or increases the weight (w_i^k) (Figure 1 (d)). The weights of behaviors provided by other agents are automatically decreased without any heartbeat messages from the agents. Therefore, when an agent terminates or fails, other agents decrease the weights of the behaviors provided by the agent and if they then have the same or compatible behaviors, they can activate the behaviors, which may be inactivated.

F. Consistent Adaptation based on Primary-Backup Protocol

Our framework uses a primary-backup scheme to maintain consistent states between one primary server and replica servers for adaptation. A primary server receives all incoming client requests, executes them, and propagates the resulting to the backup replica severs. To detect failures in the primary and replica servers, they periodically sends heartbeat messages to one another.

- When the primary server crashes, some of the replica servers detects the inactivation of the primary because they cannot receive any heartbeat messages from the primary. They execute a recovery protocol both to agree upon a common consistent state before resuming regular operation and to establish a new primary to broadcast state changes. To exercise the primary role, a replica server must have the support of a quorum of processes. As replica servers can crash and recover, there can be over time multiple primaries and in fact the same replica server may exercise the primary role multiple times.
- When replica servers crash, the primary detects the inactivation of the servers because it cannot receive any heartbeat messages from them. It removes them from its list of replica servers. When they become activated, it sends the latest updates of the state that were adapted after they crash.

We associate an instance value with each established primary. A given instance value maps to at most one replica server.

Each runtime system is constructed as a middleware system with Java (Figure 2). It is responsible for executing agents and for exchanging messages in runtime systems on other computers through a network. When a runtime system

 ISBN: 978-1-61208-391-9

is (re)connected to a network, it multicasts heartbeat messages to other runtime systems to advertise itself, including its network address in a plug-and-play protocol manner.

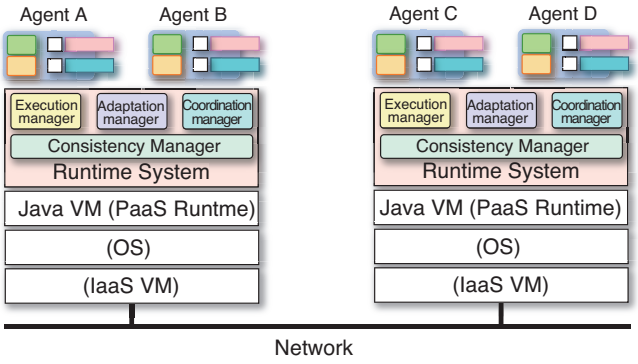


Figure 2. Runtime system

Adaptation messages, i.e., restraining and heartbeat messages, are transmitted as multicast UDP packets, which are unreliable. When the runtime system multicasts information about the signature of a behavior in restraining messages, the signature is encoded into a hash code by using Java's serial versioning mechanism and is transmitted as code. Restraining messages for behaviors that do not arrive at agents are seriously affected, because other agents automatically treat the behaviors provided by the senders to be inactive when they do not receive such messages for certain durations. Since our mechanism does not assume that each agent has complete information about all agents, it is available even when some heartbeat messages are lost.

Application-specific messages, i.e., request and reply, are implemented through TCP sessions as reliable communications. When typical network problems occur, e.g., network partitioning and node failure during communication, the TCP session itself can detect such problems and it notifies runtime systems on both sides to execute the exception handling defined in runtime systems or agents. The current implementation supports a multiplexing mechanism to minimize communication channels between agents running on two computers on at most TCP session. To avoid conflicts between UDP packets, it can explicitly change the periods of heartbeat messages issued from agents. Each runtime system offers a remote method invocation (RMI) mechanism through a TCP connection. It is implemented independent of Java's RMI because this has no mechanisms for updating references for migrating components. Each runtime system can maintain a database that stores pairs of identifiers of its connected components and the network addresses of their current runtime systems. It also provides components with references to the other components of the application federation to which it belongs. Each reference enables the component to interact with the component that it specifies, even if the components are on different hosts or move to other hosts.

V. Evaluation

Although the current implementation was not constructed for performance, we evaluated that of several basic operations in a distributed system where eight computers (Intel Core 7i Duo 2.8 GHz with MacOS X 10.9 and J2SE version 7) were connected through a giga-ethernet. The cost of transmitting a heartbeat or restraining message through UDP multicasting was 11 ms. The cost of transmitting a request message between two computers was 21 ms through TCP. These costs were estimated from the measurements of round-trip times between computers. We assumed in the following experiments that each agent issued heartbeat messages to other agents every 100 ms through UDP multicasting.

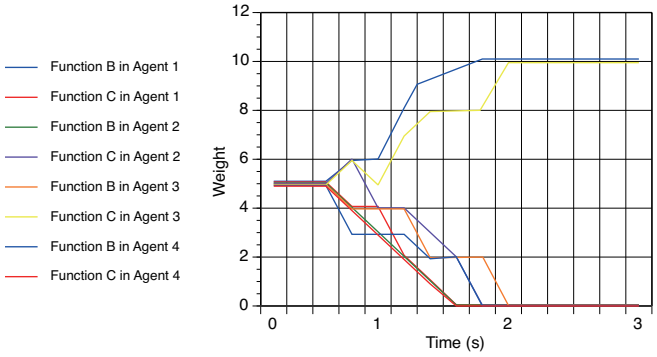


Figure 3. Degree of progress in differentiation-based adaptation

The experiment was carried out to evaluate the basic ability of agents to differentiate themselves through interactions in a reliable network. Each agent had three behaviors, called A, B, and C. The A behavior periodically issued messages to invoke its B and C behaviors or those of other agents every 200 ms and the B and C behaviors were null behaviors. Each agent that wanted to execute a behavior, i.e., B or C, selected a behavior whose weight had the highest value if its database recognized one or more agents that provided the same or compatible behavior, including itself. When it invokes behavior B or C and the weights of its and others behaviors were the same, it randomly selected one of the behaviors. We assumed in this experiment that the weights of the B and C behaviors of each agent would initially be five and the maximum of the weight of each behavior and the total maximum W^k of weights would be ten.

Figure 3 presents the results we obtained from the experiment. Both diagrams have a timeline in minutes on the x-axis and the weights of behavior B in each agent on the y-axis. Differentiation started after 200 ms, because each agent knows the presence of other agents by receiving heartbeat messages from them. Figure 3 shows the detailed results of our differentiation between four agents, where their weights were not initially varied and then they forked into progression and regression sides.

 ISBN: 978-1-61208-391-9

VI. APPLICATION

We describe a practical application with this approach to illustrate the utility of our (de)differentiation for service deployment and composition in a disaggregated computing setting. Our application is initially a single drawing editor service at a server and then automatically duplicates, deploys, and differentiates the service and its clones at different computers. Each service is defined based on a model-view-control (MVC) pattern as an an agent consisting of *model*, *view*, and *control* behaviors. The first manages and stores drawing data and should be executed on a computer equipped with a powerful processor and a lot of memory. The second part displays drawing data on the screen of its current host and should be deployed at computers equipped with large screens. The third part forwards drawing data from the pointing device, e.g., mouse, of its current computer to the first behavior.

When the server is connected to a network, the agent automatically introspects the capabilities of computing devices connected to the network via the current runtime system. When it discovers a computer equipped with a pointing device and a large display, e.g., a smart TV, the agent makes a clone of it with its behaviors and deploys the clone agent at the smart TV. The original agent, which is running on the server, decreases the weights of its behaviors corresponding to the view and control parts and the clone agent, which is running on the smart TV, decreases the weight of its behavior corresponding to the model part. This is because the server has no display or pointing device and the smart TV had no storage device. Therefore, each of the agents delegates the behaviors that its computer does not support to another agent. As a result, their weights for behaviors monotonously increases or decreases and they are then successfully differentiated according to the capabilities of their current computers. A user could view pictures stored in the server on the screen of a smart TV.

When a user disconnects the server from the network, the agent running on the server dedifferentiates itself, because it lacks its co-partners, to which it delegates the behaviors corresponding the view and control parts. When it connects to another network with another smart TV, it can clone itself and differentiate itself and the clone. However, since the agent running on the smart TV has no data, it does not invoke inactive behavior, which corresponds to the model part, and is then terminated.

Although this may be carried out by using non-differentiation approaches, this approach had several advantages. For example, it has no central management system so that it can avoid single points in performance bottlenecks and failures. It make our management tasks easier. That is, after we just deploy only one agent at a computer, the approach enables the agent to automatically duplicate, deploy, and adapt itself and its clones according to the capabilities of computers and the demands of its applications.

VII. CONCLUSION

This paper proposed a framework for adapting software agents, which coordinate with one another, on distributed systems. It is unique to other existing software adaptations in introducing the notions of differentiation and cellular division in cellular slime molds, e.g., dictyostelium discoideum, into software agents. When an agent delegates a function to another agent, if the former has the function, its function becomes less-developed and the latter's function becomes well-developed. When agents have many requests from other agents, they create their clone agents. The framework was constructed as a middleware system on real distributed systems instead of any simulation-based systems. Agents can be composed of Java objects.

REFERENCES

[1] G. S. Blair, at al., Reflection, self-awareness and self-healing in OpenORB, in Proceedings of 1st Workshop on Self-healing systems (WOSS'2002), ACM Press, 2002, pp.9–14.

[2] S. Cheng, D. Garlan, B. Schmerl, Architecture-based self-adaptation in the presence of multiple objectives, in Proceedings of International Workshop on Self-adaptation and Self-managing Systems (SEAMS'2006), ACM Press, 2006, pp.2–8.

[3] I. Georgiadis, J. Magee, and J. Kramer, Self-Organising Software Architectures for Distributed Systems in Proceedings of 1st Workshop on Self-healing systems (WOSS'2002), ACM Press, 2002, pp.33–38.

[4] M. A. Jaeger, H. Parzyjegla, G. Muhl, K. Herrmann, Self-organizing broker topologies for publish/subscribe systems, in Proceedings of ACM symposium on Applied Computing (SAC'2007), ACM, 2007, pp.543–550.

[5] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, 1992.

[6] W. Lin, G. Lin, and H. Wei, Dynamic Auction Mechanism for Cloud Resource Allocation In Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid'2010), 2010, pp.591–592.

[7] T. Nakano and T. Suda, Self-Organizing Network Services With Evolutionary Adaptation, IEEE Transactions on Neural Networks, vol.16, no.5, 2005, pp.1269–1278.

[8] I. Satoh, Evolutionary Mechanism for Disaggregated Computing, In Proceedings of 6th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS'2012), IEEE Computer Society 2012, pp.343–350.

[9] T. Suda and J. Suzuki: A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications. IEEE Journal on Selected Areas in Communications, vol.23, no.2, 2005, pp.249–260.

[10] P. L. Snyder, R. Greenstadt, G. Valetto, Myconet: A Fungi-Inspired Model for Superpeer-Based Peer-to-Peer Overlay Topologies, in Proceedings of 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO'2009), 2009, pp.40–50.

[11] Q. Zhang, E. Gurses, R. Boutaba, and J. Xiao, Dynamic resource allocation for spot markets in clouds, in Proceedings of 11th USENIX Conference on Hot topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE'2011), USENIX Association, 2011.

 ISBN: 978-1-61208-391-9

Adaptive Experience-Based Composition of Continuously Changing Quality of Context

Mats Neovius
Department of Computer Science
Åbo Akademi University
Turku, Finland
e-mail: mneovius@abo.fi

***Abstract*—Contemporary systems increasingly rely on information provided by autonomous agents. The autonomous agents provide inherently inaccurate information due to, for example, rounding, calibration error or subjectivity. Moreover, the level of information inaccuracy may change without notice. Regardless the reason of the inaccuracy, a system relying on such information needs to adapt to the quality of the momentary information. In this paper, we propose a method for this adaption. The method bases on evidence theory and probability theory to compose a ground truth from disjoint information in a proposition. This ground truth is used to evaluate the disjoint information and determine this experience's score. Each experience adds to the history of experiences in an agent, i.e., to the amount and character of evidence an agent has on another's performance. Moreover, the method features a forgetting parameter facilitating adaption in case of, for example, maintenance to the providing agent. The method output is one parameter denoting the level of confidence the system certifies the composed information with. The presented method is validated by a case study on a dataset of in-house temperature data.**

Keywords-Experience-based trust; adaptive systems, reputation-based trust, evidence theory; uncertainty, trust model.

I. INTRODUCTION

In the era of big data and cloud computing, a system manifests in an application providing a human user a means to perform a task [1 – 2]. The device, in this case, functions as the mere portal to the user's application space and information enhanced environment; with the application providing the user-interface and the gateway to the Internet scale information enhanced environment. This information enhanced environment is stored, maintained, updated and provided by autonomous agents connected 'all the time everywhere' [3]. Hence, the application produces automated transactions in the Information Revolution [4] without any conception of the momentary level of confidence that may justifiably be placed on the information it relies on.

In this setting, each piece of information is subject to the context in which it was created. For example, consider a measurement value of an elementary sensor, the information imperfection includes: imprecision when being inexact, ambiguity when non-unanimous data are available, error when a mismatch is found between the actual and reported value and unknown when the properties are not fully known [5]. On these, van Bunningen et al. [6] note that information dependent on the context of its measurement is continuously changing, imperfect and *uncertain*. In wireless sensor networks, Nakamura et al. [7] list reasons including variations of temperature, pressure, electromagnetic noise, radiation and conclude that sensors' measurements may be rendered imprecise (or even useless). These inaccuracies and ambiguities, being the context of the information, count for the inherent inaccuracy that all contextual information is subject to.

This inaccuracy is captured by the concept of Quality of Context (QoC) [8]. In QoC, context is defined as "*…any information that can be used to characterize the situation of an entity…*" [9] and it describes the contextual information's distance from the real world [10]. QoC is defined on the information, not the provider, with parameters including: precision, probability of correctness, *trustworthiness*, resolution and up-to-dateness [8]. Of these, trustworthiness is noted as the rated certainty of the other QoC parameters. That is, trustworthiness is the information providing agent's level of certification on the information it provides. Thus, the parameter of trustworthiness outlines a level of confidence and (un)certainty in the information.

McKnight and Chervaney [11] define the trusting intension as "*The extent to which one party is willing to depend on the other party in a given situation with a feeling of relative security, even though negative consequences are possible*". From this definition we note that the parties are *agents*, called *trustor* and *trustee*, where a trustor willingly trusts a trustee despite a risk of a negative outcome. Thus, trust is valid only when something can go wrong. Moreover, as the feeling of relative security may vary by trustor and situation, the level of trust is subjective. With respect to confidence, (un)certainty and the ever changing context, we further stress that a level of trust needs to continuously adapt. For this, we use Dempster-Shafer (DS) theory of evidence and its extension called Subjective Logic (SL).

SL is a probabilistic logic originally proposed by Jøsang [12]. It may be used to analyse Bayesian networks [13]. SL provides a computational logic for calculating subjective trust by a three-valued parameter called an opinion. Moreover, it features second-hand evidence. A mapping function between

 ISBN: 978-1-61208-391-9

a SL opinion and the Beta probability density function (Bpdf) have been devised [12] [14]. To provide the input to SL and Bpdf, we outline an *experience* in line with Teacy [15] and Neovius [16] as a four tuple. Each experience is a piece of evidence of an agent's evaluated transaction with another agent. The set of experiences is the history of an agent's evaluated transactions. Hence, the *experience-based* trust; sometimes used interchangeably with the concept of reputation-based.

In this paper, we outline the logics of a trust calculation and present a computational model for adaption by trust levels. The model's output is a tuple, called the abstract score that complies with the Bpdf type and thus, with SL framework. We extend on previous work [17] by the generality, motivating and presenting the mathematical treatment of uncertainty in DS theory and by highlighting the adaptive behaviour. Moreover, we elaborate on the history of experiences; motivating the experience-based approach. This is also the main contribution of this paper; as the history of experiences builds up and decays, the level of trust adapts. Based on the findings, we claim that the adaptive behaviour enables correction of inherently inconsistent and inaccurate information. To the best of our knowledge, this paper is the first to study the combination of context, adaption and subjective trust from a mathematical-logical point of view. In addition, this paper elaborates on the limitations of the approach by presenting assumed properties of a trust relation. Thus, this paper takes a more general view on experience-based trust and uses the in-house case study as validation; as opposed to the case study walk-through provided in previous work [17]. The most severe limitation of the approach is the need of a ground truth. If such a ground truth cannot be devised, the method is void. However, a ground truth may be defined by a human evaluation, derived indirectly or, as in the case study, derived as a variant of redundant measurements.

The general plot of this paper is depicted in Fig. 1 featuring three kinds of agents, inspired by related work [18] - [22]. Top left agent observes a property of a phenomenon transforming a real world event to a software event. Realistically, this is a temperature sensor. Bottom left, in the middle and bottom right are agents that rely on observations and other providing agents for supplying information needed for inference. The inference may, or may not give rise to triggering an actuator (top right) transforming the software event back to a real world event. We note, however, that the inference is out of the scope of this paper.

The reminder of this paper is organized as follows: In Section II, we outline the basics for evidence theory. Section III describes the experience-based trust including the level of trust, experiences, score type, decay function and a means to abstract the history of experiences to outline a subjective level of trust. Section IV describes the case study and highlights a clear point of adaption. Section V concludes the paper followed by references in Section VI.

II. DEMPSTER-SHAFER THEORY OF EVIDENCE

As noted, a level of trust encompasses the level of confidence as dependence and reliance and a level of (un)certainty. For this, DS theory of evidence [23], also known as a belief function fits well. A belief function relies on a set of known outcomes ζ, called the frame of discernment. This frame denotes the exclusive and exhaustive outcomes, e.g., in case of determining the colour of a ball, all possible colours. On a frame, the mass (certainty) $m: 2^{\zeta} \rightarrow [0,1]$ denotes the level of evidence for each outcome. The probabilistic view on the evidence assigns *m* to each element 2^{ζ} and is called basic belief assignment where $m(\emptyset) = 0$ and $\sum_{A \in 2^{\zeta}} m(A) = 1$. This additivity denotes that an evaluation is performed each time. In case the observer is uncertain, e.g., in case a red-green colour blind person evaluates a red ball on mass space $\zeta = \{x_{red}, x_{green}, x_{blue}\}$ the evaluation is $(\{x_{red}, x_{green}\})$. That is, the evaluation provides a piece of evidence for "not x_{blue}".

In addition to the mass *m*, the belief *bel* is defined $bel(A) = \sum_{B \subseteq A} m(B)$. Hence, *bel* denotes the 'certainty' or 'evidence' in a set as the sum of masses of its subsets, e.g., $bel(\{x_1, x_2\}) = m(\{x_1\}) + m(\{x_2\}) + m(\{x_1, x_2\})$. The mass of the total set $m(\zeta)$ may not be 0, i.e., $m(\{x_1, x_2, x_3\}) \neq 0$. Realistically, this is the case when a blind person would evaluate a ball's colour. Plausibility *pl* denote the 'max probability'; or that 'there is evidence against this proposition'. Thus, $pl \geq bel$ and $pl(A) = \sum_{A \cap B \neq \emptyset} m(B)$; the sum of non-empty intersecting masses or more conveniently, $pl(A) = 1 - bel(\bar{A})$ where $\bar{A}$ denotes complement of *A*. Thus, belief and plausibility provides the

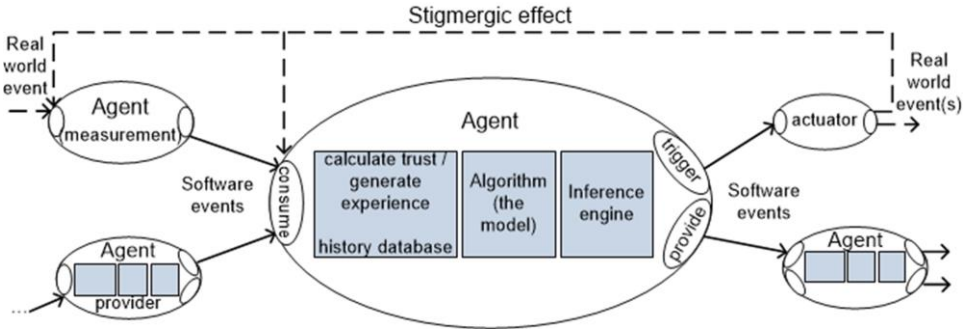


Figure 1. The adaptive agent

lower (*bel*) and upper (*pl*) bounds of evidence with the *uncertainty* as the difference between these. Consequently, DS theory provides a foundation for evidence-based trust. Readers interested in this relation and its theoretical foundations with trust are directed elsewhere [24] [25].

III. EXPERIENCE-BASED TRUST

An experience is an evaluation generated by a trustor *A* on a transaction it had with a trustee *B*. Obviously, when the evaluation is subject to bias or appreciation, *A*'s evaluation is *subjective*. This implies that if agents *B* and *C* share an experience on a matter, *A*'s and *C*'s evaluations may justifiably be different without anyone "lying". Moreover, *A*'s trust in *B* does not indicate anything of *B*'s trust in *A*, hence trust is *asymmetric*. On the history of experiences, it is motivated that more recent experiences weighs heavier. Hence, the level of trust is *incomplete*, i.e., it is non-absolute and non-dogmatic. This implies that trust *evolves over time* and is non-monotonic. Non-monotonicity fundamentally differentiates experience-based trust from statistical model checking methods. Moreover, as *B* may provide information regarding disjoint frames, e.g., observing the colour of a ball and its elasticity, the trust level is *proposition specific*. That is, given disjoint ζ_1 and ζ_2, *A*'s level of trust on *B* in ζ_1 and ζ_2 may be different [26]. The proposition specificity also encapsulates a frame of discernment from other frames, thus, voiding cross-layer effects of unforeseen dependencies.

The property of trust transitivity is discussed in literature [13] [27] - [29] with motivations for and against. In the presented method, positive trust is considered transitive, but negative trust (distrust) is not [30]. That is, if agent *A* trusts *B* and *B* trusts *C*, then by transitivity *A* trusts *C*. If distrust were transitive solving whether your enemy's enemy is your friend [25] would be required, i.e., if *A* distrust *B* and *B* distrust *C*, does this indicate that *A* trusts *C*? This motivates our view that distrust indicates not to trust any information provided by a distrusted agent, here *B*. More on these trust properties and their foundations is found elsewhere [27].

A. The Level of Trust

Let a trust relation derived from experiences between two agents, *A* and *B*, be denoted by $_{\mathrm{T}}$. Moreover, let the level of trust be denoted ω. Thus, agent *A*'s trust in *B* in a proposition is denoted $A_{\zeta T\omega}B$. Whenever the proposition $\zeta \subset 2^{\zeta} \wedge \zeta \neq \emptyset$, this level is subadditive. A subadditive level of trust features the levels of confidence as (dis)belief and (un)certainty. Here, uncertainty must not be confused with (dis)belief, also known as distrust [31] [32]. Moreover, the uncertainty enables implementation of decay reducing evidence without subverting the experience.

In addition to the level, we distinguish between first-hand and second-hand trust levels as in SL [12]. A first-hand trust level is derived from first-hand direct (*d*) experiences with the trustee in a proposition. A second-hand trust level is an indirect (*i*) level, where referral agents' experiences are used to strengthen an agent's evidence. Moreover, trusting an agent as a referral is a meta proposition in its own right; thus we consider trust either referral (*r*) or functional (*f*). Indirect functional trust when $A_{\zeta Tdr\omega}B$ and $B_{\zeta Tdf\omega}C$ between agent *A* and *C* is denoted $A_{\zeta Tif\omega}C$, depicted in Fig. 2.

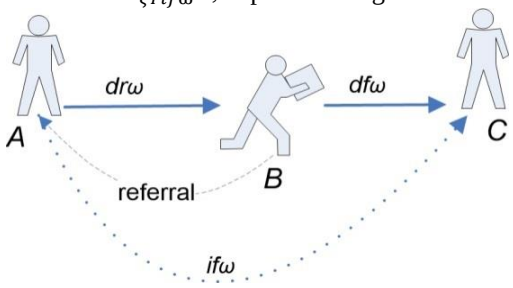


Figure 2. Trust transitivity

B. The Experiences

In order to derive a level of experience-based trust, we need to define the experience type. The type is defined a four tuple $(\delta, \epsilon, \zeta, \eta)$, inspired by Krukow [33] Teacy et. al [15] first introduced in Neovius [16]. The elements of the tuple are: $\delta \in \{\langle agents \rangle\}$, ϵ as the datum, $\zeta \subseteq \{\langle frame \rangle\}$ and $\eta \in \{\langle score \rangle\}$. Realistically, the datum ϵ is time. An experience generated by agent *A* on *B* at *x* in proposition *y* with score *z* is denoted $Exp^A(\epsilon) = (B, x, y, z)$. The history of agent *A*'s experiences $Exp^A(\epsilon_i)$ for *i* = 1, …, *n* is a set of disjoint experience, i.e., $\{(\delta, \epsilon, \zeta, \eta)\}$. Adding a new experience $(\delta, \epsilon_0, \zeta, \eta)$ at datum ϵ_0 to the history is straight forward $Exp^A(\epsilon_j) = (\delta, \epsilon_0, \zeta, \eta) \cup \{(\delta, \epsilon_i, \zeta, \eta)\}$ where *j* = 1, ..., *n*, ϵ_0.

On this experience type, projections provide specific experiences. Consider agent *A* to have interacted with *B*, then experiences on *B* at ϵ are projected by $Exp^A(B, \epsilon) = \{(B, \epsilon, \zeta, \eta)\}$ where $\{(B, \epsilon, \zeta, \eta)\} \subseteq \{(\delta, \epsilon_i, \zeta, \eta)\}$. Projecting on any element is done similarly, e.g., $Exp^A(B, \epsilon_p, x) = \{(\epsilon_i, \eta)\}$ for $x \subseteq \zeta$ and $\epsilon_i \leq \epsilon_p$ for *i* = 0, 1, … *p*.

C. Type of Score

We propose a versatile score type as a tuple (*sat*, *usat*) for satisfactory and unsatisfactory. Here $sat, usat \in [0, 1]$ and $sat + usat \leq 1$. Subadditivity is fundamental for implementing uncertainty and decay without subverting the semantics of the experiences' score. A score is dogmatic whenever $sat + usat = 1$. Coarsening a multinomial proposition $|\zeta| \geq 3$ to a binomial $|\zeta| = 2$ is done by summation, i.e., considering the coloured balls $\zeta = \{x_r, x_g, x_b\}$ with $Exp^A\left(A, \epsilon, (x_r, x_g)\right) = \{(\epsilon_i, (sat, usat))\}$ providing the evidence against x_b. Related work considering a similar score type includes Noorian et al. [34] model.

With this score type, vacuous experiences are expressed with the score (0, 0); dogmatic scores (the probabilistic view) $sat + usat = 1$; and absolute scores (binary view) when (*sat*, *usat*) = (0, 1) or (*sat*, *usat*) = (1, 0). Thus, a score is valid with a certainty of *sat* + *usat*, e.g., given *sat* = 0.3 and *usat* = 0.5, the certainty is 0.8. From this, uncertainty *u* is easily

derived, $u = 1 - sat - usat$ as is the dogmatic expectation value of satisfyability as $sat\ /\ (sat\ +\ usat)$.

D. Decay of an Experience Score

Decay of an experience score is the act of forgetting or forgiving. This is fundamental in case of transient faults or maintenance / update on the data provider having an effect on the data quality. Fundamental for decay is that it must not subvert the score of experiences, only reduce its weight [34]. Let the decay factor be denoted by λ where $0 \leq \lambda \leq 1$. Decay relies on a continuous factor by which it decays, here datum ϵ. We define the general decay function d at datum ϵ_m called d_{ϵ_m} on an experience $Exp^{\delta}(\epsilon_i)$ where $\epsilon_i \leq \epsilon_m$ as:

$$d_{\epsilon_m}\left(Exp^{\delta}(\epsilon_i)\right) = (\delta', \epsilon_i, \zeta, \lambda^{\epsilon_m - \epsilon_i} * \eta) \quad (1)$$

Dually, this decay may be applied on the history of experiences where $\epsilon_n = 1, \ldots, m$ and $\epsilon_n \leq \epsilon_m$:

$$d_{\epsilon_m}\left(Exp^{\delta}(\epsilon_n)\right) = \{(\delta', \epsilon_n, \zeta, \lambda^{\epsilon_m - \epsilon_n} * \eta)\} \quad (2)$$

With equations (1) and (2), the closer λ is to 1 the slower the speed of decay with $\lambda = 1$ indicating no decay at all. No decay is motivated in, among others, quantitative statistical analysis. Contrary, $\lambda = 0$ indicates complete decay as in a stochastic process [35].

E. Abstracting Experiences

To calculate with the experiences, the projection on the experiences' scores needs to be composed. We call this an abstract experience *Abs* at datum ϵ_m, hence Abs_{ϵ_m}. If this abstraction is done on decayed experiences, such a composition outlines the momentary decayed score, the abstracted score *absscore*. We define this for ϵ_n = 1, …, m and $\epsilon_n \leq \epsilon_m$:

$$Abs_{\epsilon_m}\left(Exp^{\delta}(\epsilon_m)\right) = (\delta', \epsilon_m, \zeta, \sum_{d_{\epsilon_m} Exp^{\delta}(\delta', \epsilon_n, \zeta)} \eta) \quad (3)$$

With this, $Abs_{\epsilon_m}\left(Exp^{\delta}(\delta', \epsilon_m, \zeta)\right)$ score *absscore* $\in \mathbb{R}^+$ relies on summation defined (*abssat*, *absusat*).

Not surprisingly, as $Abs_{\epsilon_m}\left(Exp^{\delta}(\epsilon_i)\right)$ denotes the *absscore* decayed at datum ϵ_m, an updated abstract view $Abs_{\epsilon_{m'}}\left(Exp^{\delta}(\epsilon_i)\right)$ where $m \leq m'$ is a recursive function [36] whenever the decaying factor is universal, continuous and applied on all experiences locally, e.g., decay by time. Hence, updating $Abs_{\epsilon_m}\left(Exp^{\delta}(\epsilon_i)\right)$ to $\epsilon_{m'}$ where $\epsilon_{m'} = \epsilon_m + i$ is straightforward:

$$Abs_{\epsilon_{m'}}\left(Exp^{\delta}(\epsilon_i)\right) = \delta', \epsilon_{m'}, \zeta, \left(Exp^{\delta}(\delta', \epsilon_{m'}, \zeta, \eta) + Abs_{\epsilon_m}\left(Exp^{\delta}(\delta', \epsilon_m, \zeta)\right) * \lambda^i\right) \quad (4)$$

Here, $Exp^{\delta}(\delta', \epsilon_{m'}, \zeta, \eta)$ is the new experience. Thereby, abstraction is an irreversible function that provides some level of privacy that decay enhances on. This abstracted experience with a score (*abssat*, *absusat*) may be depicted as a Bpdf and is, hence, mappable to an opinion in SL. Examples may be found elsewhere [36] - [38]. Computationally, the method is very light thus, facilitating scalability.

IV. In-house Case Study

As proof of concept, we have applied the presented method on an in-house temperature measurement system. This system encompasses a dataset of four disjoint points of temperature measurement with 10 seconds interval over a time span of one year; a total of ~12 million readings. We filtered the dataset from impossible measurements such as -49950°C by disregarding readings outside the interval $[-50°C, 50°C]$. We used $\lambda = 0.95$ as in eq. (1, 2, 4).

The purpose is to model an "in-house temperature agent" that composes the disjoint measurements to the most probable temperature and certifies this by a level of trust. For this, the method provides a weighted mean temperature (*wmt*) and a weighted level of trust (*wlt*). The wlt defines the momentary level of trust that the in-house temperature agent certifies the wmt with that sets the ground truth. An elementary temperature measurement experience's score is generated by the three-sigma rule of standard deviation from the normal distribution of the posterior wmt. Thus, initially with no experiences (equal trust on all measurements) the wmt is the arithmetic mean.

A snapshot of the analysis is depicted in Fig. 3. The abbreviations in the legend of Fig. 3 are as follows. On the left scale: FiPl = fireplace sensor, LiRo = living room sensor, wlt, Hallway = hallway sensor, BedR2ndF = bedroom 2nd floor sensor; and on the right scale: Mean temp. = arithmetic mean temperature in °C, Daily mean = the daily mean temperature outdoor in °C and wmt in °C. The primary vertical axis denotes the trust level, the secondary vertical axis denotes the temperature °C, and the horizontal axis denotes time as mm.dd.yyyy hh:min.

Temperature	Measurements (4000sec. interval) in °C						
	Arith. mean	***LiRo temp***	***EnWa temp***	***B2F temp***	***FiPl temp***	***wmt***	***wlt***
6.15.2013 3:13 –3:46	21,68	23,55	22,75	23,26	17,18	23,19	0,72
	26,40	23,55	22,42	23,26	36,37	23,08	0,79
	17,77	23,55	22,42	23,26	1,8	23,08	0,80
	18,82	23,55	22,42	23,26	5,67	23,08	0,81
	21,60	23,55	22,42	23,26	17,18	23,08	0,80
	21,64	23,55	22,55	23,26	17,18	23,12	0,81
…	…	…	…	…	…	…	…
6.27.2013 13:37 – 6.27.32013 14:26	26,65	29,35	26,36	29,88	21,02	28,59	0,77
	28,29	23,43	26,36	27,01	36,37	25,64	0,76
	24,45	23,43	26,36	27,01	21,02	25,67	0,75
	24,44	23,43	26,29	27,01	21,02	25,66	0,75
	24,45	23,43	26,36	27,01	21,02	25,69	0,71
	21,56	23,43	26,29	27,01	9,51	25,69	0,66
	21,58	23,43	26,36	27,01	9,51	25,69	0,74
	23,48	23,43	26,29	27,01	17,18	25,68	0,74

TABLE I. A SAMPLE OF TEMPERATURES AND TRUST LEVELS

Fig. 3 reveals that FiPl is malfunctioning by a close to 0 level of trust. Table 1 depicts this inconsistency as a more specific snapshot. Fig. 3 also reveals that once the outdoor temperature (daily mean) exceeds approximately 19°C, the trustworthiness levels start to deviate. This holds true when inspecting the raw data, with the conclusion that LiRo and

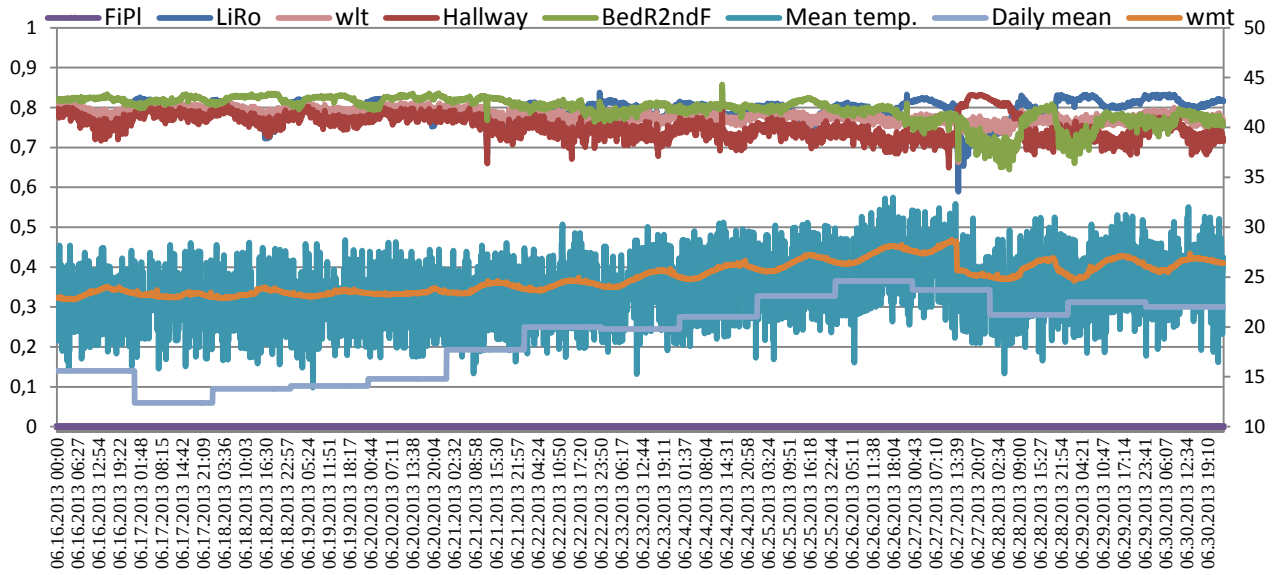


Figure 3. Indoor temperature and measurement processes' trust levels with λ = 0.95 (June 2013)

BedR2ndF correlates and vary more heavily depending on the time of day and outdoor temperature, whereas Hallway is more stable. To illustrate this, Fig. 3 plots a timespan of late June 2013, when the outdoor temperature at the location of the house was higher than the adjusted indoor temperature resulting in deviations in the levels of trust.

Notable in the graph is the sudden drop of trust levels of LiRo, BedR2ndF and the relative increase of Hallway points of measurement 06.27.2013 at around 13:40. The reason for this is a thunder storm. Fig. 3 plots this as a drop in wmt and inconsistency in trust levels in a reasonable manner only to recover gradually, with a lower λ the recovery is faster. Hence, the system adapted to the change in information quality. Moreover, it reasonably dropped the weighted mean temperature and the weighted mean trust during the inconsistent event of the thunder storm.

V. CONCLUSIONS

The trend in contemporary computerised systems is towards agent and system autonomy. Concepts like system of systems, software as a service and many alike are proofs of this. In all these cases, the application performing a task for a stakeholder is assumed to rely on information provided by agents not in its control. As there is no guarantee on the providing agents' reliability, a consuming agent may only adapt to the momentary best-effort perception on the providing agent. A survey may be found elsewhere [39]. This paper motivates, define and use a level of trust as the basis for adaption.

The approach is implemented on a dataset of four temperature sensors. Though this dataset is very well fitted for this particular approach, the underlying mathematics is described in detail sufficient to be applied on related problem scenarios. We believe that the results will be good if done; an issue that remains as future work. Moreover, the method scale as it is computationally light. In addition, the case-study sensors could be replaced by an adaptive agent as in Fig. 1, e.g., being provided by a service.

ACKNOWLEDGMENT

This research is funded by the Academy of Finland project "FResCo: High - quality Measurement Infrastructure for Future Resilient Control Systems" (Grant nr. 264060).

REFERENCES

[1] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman and D. Zukowski, "Challenges: an application model for pervasive computing.," In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking, 2000.

[2] G. Banavar and A. Bernstein, "Software infrastructure and design challenges for ubiquitous computing applications," Commun. ACM, vol. 45, nr 12, pp. 92-96, 2002.

[3] S. Marzano and E. Aarts, The New Everyday View on Ambient Intelligence, Uitgeverij 010 Publishers, 2003.

[4] H. Alesso and C. Smith, Thinking on the web, Wiley Inc. ISBN-13: 978-0-471-76814-2, 2006.

[5] K. Henricksen and J. Indulska, "Modelling and Using Imperfect Context Information," In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04), 2004.

[6] A. van Bunningen, L. Feng and P. Apers, "Context for ubiquitous data management," International Workshop on Ubiquitous Data Management, 2005.

[7] E. Nakamura, A. Loureiro and A. Frery, "Information fusion for wireless sensor networks: Methods, models, and classifications," ACM Comput. Surv. , vol. 39, nr 3, 2007.

 ISBN: 978-1-61208-391-9

[8] T. Buchholz, A. Küpper and M. Schiffers, "Quality of Context Information: What it is and why we need it," In proc. of the 10th HPOVUA workshop , 2003.

[9] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," Workshop on the What, Who, Where, When, Why and How of Context-Awareness, 2000.

[10] C. Villalonga, D. Roggen, C. Lombriser, P. Zappi and G. Tröster, "Bringing quality of context into wearable human activity recognition systems.," In Proceedings of the 1st international Conference on Quality of Context, 2009.

[11] H. McKnight and N. Chervaney, "The Meanings of Trust," Technical Report Working Paper Series 96-04, 1996.

[12] A. Jøsang, "Artificial Reasoning with Subjective Logic," Second Australian Workshop on Commonsense Reasoning, 1997.

[13] A. Jøsang, R. Hayward and S. Pope, "Trust network analysis with subjective logic," In Proceedings of the 29th Australasian Computer Science Conference, 2006.

[14] A. Jøsang, "Trust-Based Decision Making for Electronic Transactions," Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99), 1999.

[15] W. Teacy, J. Patel, N. Jennings and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," Autonomous Agents and Multi-Agent Systems, vol. 12, nr 2, pp. 183-198. , 2006.

[16] M. Neovius, "Trustworthy Context Dependency in Ubiquitous Systems," PhD thesis, TUCS Dissertations, 2012.

[17] M. Neovius, M. Stocker, M. Rönkkö and L. Petre, "Trustworthiness Modelling on Continuous Environmental Measurement," In Proceedings of the 7th International Congress on Environmental Modelling and Software, 2014.

[18] J. Coutaz and G. Rey, "Foundations for a Theory of Contextors," Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, 2002.

[19] G. Rey and J. Coutaz, "The Contextor Infrastructure for Context-Aware Computing," Component-oriented Approaches to Context-aware Computing ECOOP'04, 2004.

[20] P. Gray and D. Salber, "Modelling and Using Sensed Context Information in the Design of Interactive Applications," In Proceedings of the 8th IFIP international Conference on Engineering For Human-Computer interaction, 2001.

[21] G. Biegel and V. Cahill, "A Framework for Developing Mobile, Context-aware Applications," In Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications (Percom'04) , 2004.

[22] A. Fitzpatrick, G. Biegel, S. Clarke and V. Cahill, "Towards a Sentient Object Model," Workshop on Engineering Context-Aware Object Oriented Systems and Environments (OOPSLA/ECOOSE'02), 2002.

[23] A. Dempster, "Upper and lower probabilities induced by a multivalued mapping," The Annals of Mathematical Statistics , vol. 38, nr 2, p. 325–339, 1967.

[24] A. Jøsang, "Subjective Logic", Draft book available http://folk.uio.no/josang/papers/subjective_logic.pdf , visited 12.2.2015.

[25] A. Jøsang and S. Pope, "Dempster's Rule as Seen by Little Colored Balls," Computational Intelligence, vol. 28, nr 4, pp. 453-474, 2012.

[26] R. Falcone and C. Castelfranchi, Social trust: a cognitive approach, C. C. a. Y. Tan, Red., In Trust and deception in virtual societies, Kluwer Academic Publishers, 2001, pp. 55-90.

[27] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," IEEE Communications Surveys and Tutorials, vol. 3, nr 4, 2000.

[28] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," In Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling , 2005.

[29] B. Christianson and W. Harbison, "Why isn't trust transitive?," In Proceedings of the Security Protocols International Workshop, 1996.

[30] T. DuBois, J. Golbeck and S. A., "Predicting Trust and Distrust in Social Networks," In IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and IEEE Third Int. Conference on Social Computing, 2011.

[31] M. Carbone, M. Nielsen and V. Sassone, "A Formal Model for Trust in Dynamic Networks,", BRICS Report Series Publications. RS-03-4, 2003.

[32] S. Marsh, "Formalizing Trust as a Computational Concept," PhD thesis, University of Stirling, 1994.

[33] K. Krukow, "Towards a theory of trust for the global ubiquitous computer,", PhD thesis, Uni. of Aarhus 2006.

[34] Z. Noorian, S. Marsh and M. Fleming, "Multi-layer cognitive filtering by behavioral modeling," In The 10th International Conference on Autonomous Agents and Multiagent Systems, 2011.

[35] A. Whitby, A. Josang and J. Indulska, "Filtering out unfair ratings in bayesian reputation systems," Proceedings of the Third International Joint Conference on Autonomous Agenst and Multi Agent Systems, 2004.

[36] A. Jøsang and R. Ismail, "The beta reputation system," In Proceedings from the 15th Bled Conference on Electronic Commerce, 2002.

[37] V. Cahill, E. Gray, J. Seigneur, C. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Serugendo, C. Bryce, M. Carbone, K. Krukow and M. Nielsen, "Using Trust for Secure Collaboration in Uncertain Environments," IEEE Pervasive Computing vol. 2, nr 3, pp. 52-61, 2003.

[38] L. Mui, M. Mohtashemi and A. Halberstadt, "A Computational Model of Trust and Reputation for E-businesses," In Proceedings of the 35th Annual Hawaii international Conference on System Sciences Hicss, 2002.

[39] A. Jøsang, R. Ismail and C. Boyd, "A survey of trust and reputation systems for online service provision," Decis. Support Syst., vol. 43, nr 2, pp. 618-644, 2007.

 ISBN: 978-1-61208-391-9

Creating a Connectivity Decision Support System for Long-Term Evolution Direct (LTE Direct)

Michael R. Bartolacci
Information Sciences and Technology
Penn State University - Berks
Reading, USA
email: mrb24@psu.edu

***Abstract*—With the advent of hybrid mobile phone network architectures where handsets can act as an intermediate point as well as a source and destination for traffic, a system is needed to allow mobile users to find "connectivity". This connectivity can be in the form of a mobile network tower, or it may be another mobile user if communicating in ad hoc mode. This work builds upon the notion of a Connectivity Decision Support System (CDSS), first proposed by the author in the context of a 1G mobile network architecture. This work in progress explores Long-Term Evolution Direct (LTE Direct) and the factors related to implementing a CDSS.**

Keywords-Connectivity Decision Support System, Long-Term Evolution Direct

I. INTRODUCTION

The ability to "stay connected" with a wireless device is an important factor for mobile users both in terms of their choice of networks and also in terms of their degree of utilization of mobile devices in the first place. Areas with poor mobile network coverage are not crowded with wireless device users fighting for limited data speeds or poor call quality. This work-in-progress proposes the application of a Connectivity Decision Support System (CDSS) to Long-Term Evolution Direct (LTE Direct), an emerging standard for hybrid mobile networks. The contribution of this work is that it examines the application of a novel system for dealing with connectivity in a wireless environment, a CDSS, as a solution for providing wireless users with useful connectivity information in the context of a hybrid network architecture (LTE Direct) where mobiles nodes can act as both sources and intermediate nodes for network traffic. In section 2, previous related work is detailed including the motivation for such a system. Section 3 examines potential design aspects for a CDSS in the context of today's mobile network architectures. In section 4, LTE Direct is described in the context of the proposed system. Section 5 examines factors that must be taken into account for the development of a CDSS for LTE Direct. The final section examines the contribution and ongoing work.

II. MOTIVATION AND PREVIOUS WORK

As more and more users rely solely on their smart phones and similar wireless devices for navigation, business, and social uses, having a fast and reliable data connection (or in the case of emergencies, voice connection) are of the utmost importance. Wireless network operators do offer connectivity maps that supposedly show areas of high and low connectivity (and ranges of service), but anyone who has viewed these maps can attest to the fact that their granularity is too large for practical purposes in the context of changing positions to gain better connectivity. They only provide rough outlines of network base station coverage. Mobile network operators therefore tend to overstate their coverage in marketing their service through these graphical depictions since smaller localized factors, especially involving manmade obstacles to theoretic signal propagation, are not truly taken into account in the maps. A CDSS, therefore, would be far more useful to wireless users seeking to improve their connectivity. In the context of battlefield networks, emergency networks, and similar wireless network implementations where maintaining connectivity is critical, a CDSS becomes almost a necessity.

The author first proposed the idea of a mechanism for guiding wireless network users to areas of greater "connectivity" in 2000, prior to the advent of "smart phones" with advanced processing and memory capabilities capable of utilizing location-based information. This system, labeled a "connectivity decision support system" at the time, would vary in exact design between various types of wireless networks, but would have the main functionality of providing mobile users with connectivity measures and information about their immediate surrounding allowing them to make informed decisions regarding their location and their connectivity with respect to mobile network connection points. The general idea of such a system was further defined in a conference paper and presentation in 2003 [1] and subsequent published work in 2004 [2]. The ability to pinpoint a mobile user's location is taken for granted today; but in the year 2000, this was a novel idea that allowed for the development of many possible network services and improvements such as a CDSS.

 ISBN: 978-1-61208-391-9

The author, along with his collaborators, were able to establish a mechanism for reliably predicting connectivity for 1G Advanced Mobile Phone System (AMPS) type networks where connectivity was defined as the received signal-to-noise (S/N) ratio for the mobile network handsets of the time period using neural networks [3]. The proposed crude form of a CDSS at that time relied upon S/N information collected from handsets to train a neural network to predict S/N ratios for all points on a given grid or region. Thus the collection of "connectivity" information from a select group of mobile users was used to create the knowledge base for advising all mobile users in a given region. At the time of this work, more sophisticated handsets with graphical user interfaces and advanced data networking capabilities were in their infancy.

This discussion of previous work related to the general idea of a CDSS has two goals: to introduce the reader to the concept and functionality of such a system, and to show that it was an idea ahead of its time. The 1G AMPS mobile network modeled in 2005 was at that time evolving into a 2G network architecture in most countries with more sophisticated data capabilities. The widespread implementation of more sophisticated technologies such as Digital AMPS and General Packet Radio Service (GPRS) ushered in the age of mobile applications and located-based services that are ubiquitous today. The concept of a CDSS that could be utilized for providing connectivity information to users for voice calls can have a more profound effect with respect to critical data-based applications and traffic.

III. IMPLICATIONS FOR A CDSS DESIGN TODAY

The author originally envisioned that a CDSS would have uses throughout a range of mobile network types and could be implemented in a variety of fashions. Ideally, a CDSS provides connectivity information to a wireless device user. Although the type of wireless device (and network therefore) determine the specific wireless standards utilized to create and operate the CDSS, the concept has applicability across a wide spectrum of wireless network types from 802.11x, to mobile phone networks, to ad hoc and hybrid ones. The two key operations for the creation and implementation of a CDSS are the measurement or prediction of connectivity surrounding a given wireless device user and the display of this connectivity information in a format that can be understood and quickly utilized by the user. Measuring or predicting connectivity around a given user is useless if such information cannot be readily used to decide if the user can achieve greater connectivity by changing positions or if there is a delay in the presentation of this information. By definition, the term "decision support system" represents an information system that advises, but does not make automated decisions, for a given user of the system.

It should be obvious that the implementation of a CDSS for a given network type, besides being tied to the technology involved for that network type, can vary as to its general operations with respect to the creation, storage and display of connectivity information to the user. A true handset-based CDSS would require processing and storage at the device level in order to determine connectivity, store such information, and then display it to a user. A provider-based CDSS would require that such information be determined and stored by a given network operator/provider and sent to a wireless user when requested. Also, a hybrid architecture using processing and storage capabilities both by the provider and the handset could also be used. The integration of advanced data networking capabilities in mobile phone networks has allowed for another possible hybrid architecture where a third party collects "connectivity" information from handsets (via a smart phone application) to create a connectivity database that is then accessed as requested by wireless users.

The premise of this "crowd sourced" CDSS was created by the author and a student in 2004 in a prototype website utilizing a database backend that stored S/N information (as a proxy for "connectivity") that was manually collected from a mobile handset for a grid of points around several local cellular network towers. Unlike the current smart phone applications such as OpenSignal [4] or Sensorly [5], which display connectivity measures only in areas on a map where actual information has been collected from users, the system developed by the author and his student used additional information and a algorithm to predict connectivity for all other points on the grid thus creating a complete map of anticipated connectivity for the physical region. The discussion of this previous work is to highlight one potential general design attribute for a CDSS today. Since today's cellular networks differ greatly from the analog AMPS-style networks in that several users simultaneously share a given bandwidth instead of having dedicated frequency channels, there is a need to take the "connectivity" of nearby users into account. A system that can utilize connectivity information from surrounding users and also predict such for a user as they move is a necessity in today's cellular networks. The same can be said for many other types of wireless architectures including the emerging hybrid LTE Direct standard. respect to critical data-based applications and traffic.

IV. LTE DIRECT AND CDSS IMPLICATIONS

At the time of the initial CDSS exploration, hybrid mobile networks were in use in only very specialized applications such as for battlefield networks. . A hybrid cellular network architecture would involve handsets operating both in a traditional manner where they communicate with a base station, and also in an ad hoc mode where transmissions between handsets can occur as well. A hybrid architecture would have the advantage of extending the communication range of the overall cellular network through the linking of "out-of- range" network devices with "in-range" ones to serve as intermediate relay points for transmissions. Several researchers have examined these types of networks, including Mane and Mohite [6] and Do,

 ISBN: 978-1-61208-391-9

Hsu and Venkatasubramanian [7]. Their work is related to various topological designs, services available and operating factors for hybrid architectures. Unfortunately, such research does not address a network from a user's point of view in terms of connecting and remaining connected to such a hybrid topology. Existing work assumes users connect and disconnect from the network much the same way they do from a traditional cellular phone network base station or a Wi-Fi hotspot. This assumption can have significant implications when such hybrid networks are employed for emergency and disaster management.

The author, with his collaborators, explored the use of various types of wireless networks for disaster planning and management including such hybrid networks [8][9][10][11]. The lack of standards in the past for true hybrid cellular architectures created a costly mix of proprietary systems and less than ideal implementations of existing wireless network infrastructure for disaster scenarios. Many patents already exist that deal with some facet of hybrid cellular networks and their operation; but the creation of a true hybrid network with handsets that can forward the traffic of other handsets to existing base stations is still in its infancy. The development of the LTE Direct standards and its technologies has direct impact on the application of wireless networks for critical emergency services and serving an affected populace after a disaster.

LTE Direct promises to deliver what are known as "Proximity Services" (ProSe) in an efficient fashion much the same way as the Bluetooth standard and its associated technologies have for many years with device to device connections. Unlike Bluetooth though, LTE direct would not have the practical 10 meter limitation on transmission distance. Emerging standards for LTE Direct in 3GPP R-12 allow for ad hoc communication between mobile network handsets and devices up to several hundred meters without a severe degradation in battery life. Also included in LTE Direct's emerging standards is Device-to-Device (D2D) ad hoc communication capabilities. This functionality directly points to a need for a wireless handset user to know the "connectivity" of his or her surroundings in terms of other users within range of transmission and potential paths to the fixed network infrastructure. Some of the aspects of LTE Direct favor the use of a CDSS to maintain connections in an ad hoc fashion.

A. *LTE Direct is "Always On"*

Unlike Bluetooth which is a service that can be turned on and off for a given mobile handset, the emerging LTE Direct standard is "always on". From a practical point of view, this means that the handset running this service is always in discovery mode receiving transmissions from its environment. "LTE Direct is a new and innovative device-to-device technology that enables discovering thousands of devices and their services in the proximity of ~500 m, in a privacy sensitive and battery efficient way. This allows the discovery to be 'Always ON' and autonomous, without drastically affecting the device battery life unlike other proximity solutions such as OTT based that use GPS, or BT-LE and WiFi Direct" [12]. This aspect of LTE Direct would favor the use of a CDSS since it would always have access to signals of all similarly enabled nodes around it. In other words, the more devices a CDSS can see, the more possibilities for increasing a given measure of "connectivity" within the network.

B. *LTE Direct would allow D2D communication*

Beyond ProSe, LTE Direct promises ad hoc communications for more than location-based information and services. It promises true device to device communication that would be invaluable for disaster and emergency management. In a disaster scenario where damage to base stations (or power failures) can render cellular network connectivity nonexistent or sporadic in hard hit areas, the ability to connect directly with other LTE Direct users would provide needed support for emergency responders and the affected populace. Current 4G LTE service relies on User Equipment (UE) communicating with evolved Node B's (eNB) (4G base stations or towers) where radio resources are locally managed (as opposed to 3G services where they were more centrally controlled). LTE Direct allows for communications between both a functioning eNB or another UE or both. This scenario radically changes the nature of cellular communications in that one does not have to be within range of a functioning eNB to be able to communicate with and through the network.

D2D communication within LTE Direct would utilize the Uplink (UL) resource and also allow for channel measurements that are reported to other UE's or to the eNB [13]. This is significant with respect to a CDSS in that it provides a possible mechanism for defining "connectivity" and creating the CDSS.

V. FACTORS FOR IMPLEMENTING A CDSS FOR LTE DIRECT

This leads to the crux of the ongoing work: the examination of the factors that must be taken into account for a CDSS that would be utilized in conjunction with LTE Direct. The first of these is the tracking of locations of mobile nodes and their relationship to base stations. Much the same way the locations of current mobile devices operating on cellular networks are stored for routing purposes, an exact geographic location for each node must be maintained by the fixed architecture network. This implies that nodes not directly in contact with the fixed infrastructure base stations must report their locations on an ongoing basis through intermediate mobile nodes in an ad hoc fashion. One can see that constant location reporting will create overhead traffic for both the ad hoc and fixed parts of the network architecture.

The second major factor that must be dealt with in applying a CDSS to LTE Direct is redundancy. The reporting of location information through intermediate nodes, although being done on a periodic basis, suffers from the potential for gaps in reporting due to node mobility and

 ISBN: 978-1-61208-391-9

reliability (nodes coming on and offline). If a CDSS is truly to provide reliable and usable information for a wireless user, it must be able to have correct information from a temporal point of view for mobile node locations. This requires redundancy in the reporting and a mechanism for ensuring that incorrect or outdated location information does not mislead the CDSS.

The third factor for a CDSS is the nature of the mobility of nodes. Within LTE Direct, D2D capabilities rely on nodes to be not only within transmission range of each other, but to have sufficient capacity and quality of signal to maintain a useful connection. The nature of mobility for each node plays a key role in this process. Transmission obstacles created by natural and manmade objects/geographic features, Rayleigh fading, and other phenomena create propagation anomalies that limit the connection between two mobile nodes. Mere physical proximity from a Euclidean distance point of view does not guarantee a useful connection for traffic. Direction and speed of a nearby node should be taken into account as well This necessitates an integration of geographic information, node mobility information and larger scale network-related information within a CDSS to provide the wireless user a true "picture" of the connectivity around him or her.

This leads to final major factor that must be dealt with, the actual measure of connectivity and how it is displayed to the user. Due to the fact that LTE Direct is a hybrid architecture, some nodes may only be operating only in D2D mode while others may also be in contact with fixed base stations. Clearly the nature of "connectivity" differs between the two modes of operation and this information must be clearly conveyed to users. One can envision a scenario where a user moves closer to a nearby mobile node under the mistaken impression that they are increasing their "connectivity" when they are actually moving away from a nearby fixed base station that would provide a more reliable connection.

VI. CONTRIBUTION AND ONGOING WORK

As mentioned previously, this is a work-in-progress paper highlighting research by the author into applying an idea borne out of an advance in wireless handset technology (GPS chipsets in mobile devices) that originated over ten years. This paper examined key factors currently being explored in order to apply a CDSS to a hybrid mobile network architecture such as LTE Direct. Whether provided as a native feature to wireless devices enabled for LTE Direct use, or as an add-on application, the time has arrived for the functionality of a CDSS across the spectrum of wireless networks. Current work is focusing on developing the algorithmic interpretation of "connectivity" for LTE Direct and the necessary protocols for a CDSS to be utilized under this standard.

REFERENCES

[1] M. Bartolacci, R. Whitaker, and S. Allen, "A connectivity decision support system (CDSS) model framework for wireless networks," Proceedings of the 11th International Conference on Telecommunication Systems Modelling and Analysis, October 2003, electronic proceedings.

[2] M Bartolacci, A. Konak, and R. Whitaker, "A connectivity decision support system (CDSS) for wireless networks," Asian Journal of Information Technology, vol. 3, pp. 807-814, 2004.

[3] M. Nasereddin, A. Konak, and M. R. Bartolacci, "A neural network-based approach for predicting connectivity in wireless networks," International Journal of Mobile Network Design and Innovation, vol. 1, pp. 18–23, 2005.

[4] OpenSignal. Available from: http://www.opensignal.com.

[5] Sensorly. Available from: http://sensorly.com.

[6] P. Mane and V. Homite, "Analysis of handoff techniques used for hybrid networks: cellular/wlan," International Journal of Research in Computer Science, vol. 2, pp. 45-50, 2012.

[7] N. Do, C. Hsu, and N. Venkatasubramanian, "Video dissemination over hybrid cellular and ad hoc networks," IEEE Transactions on Mobile Computing, vol. 13, February 2014.

[8] M. Bartolacci and E. Gomez, A 'virtual SONET' routing architecture for ad hoc networks in environmental monitoring and emergency management," Proceedings of the 2011 Wireless Telecommunications Symposium (WTS), April 2011, electronic proceedings.

[9] E. Gomez and M. Bartolacci, "Crisis management and mobile devices: extending the usage of sensor networks within an integrated system framework," Proceedings of the 8th International ISCRAM Conference, May 2011, Available from http:///www.iscramlive.org/ISCRAM2011/proceedings/papers/193.

[10] M. Bartolacci, A. Mihovska, and D. Ozceylan, "Optimization modeling and decision support for wireless infrastructure deployment in disaster planning and management," Proceedings of the 10th International ISCRAM Conference, May 2013, Available from: http://iscramlive.org/ISCRAM2013/files/216.pdf.

[11] M. Bartolacci, C. Aubrecht, and D. Aubrecht, "A portable base station optimization model for wireless infrastructure deployment in disaster planning and management," Proceedings of the 11th ISCRAM Conference, May 2014, Available from: http://iscram2014.ist.psu.edu/sites/default/files/misc/proceedings/p102.pdf.

[12] Qualcomm Presentation, Slide 5, Available from: https://www.qualcomm.com/invention/research/projects/lte--proximity-services.pdf.

[13] Z. Ghadialy, "Direct Communication in 3GPP", Slide 14, Available from: http://www.slideshare.net/zahidtg/direct-communication-in-3ggp5.

 ISBN: 978-1-61208-391-9

Study on Special Characteristics of Japanese Firms in Adaptive Processes of Requirements Definition

From Viewpoint of Comparison of Building Trust with Stakeholders between Japanese and U.S. Firms

Keisuke Kiritani
Graduate School of Policy Studies
Chuo University
Tokyo, JAPAN
e-mail: kiritani_keisuke@intec.co.jp

Masakazu Ohashi
Graduate School of Policy Studies
Chuo University
Tokyo, JAPAN
e-mail: ohashimac@gmail.com

***Abstract*— Requirements definition is an important work process for a project and it may determine the success or failure of system development project. If the requirements definition is not adequate, ambiguous conclusions may be drawn, which will lead directly to the failure of such a project. Language is used as the general method of compiling the requirements definition into drawings and documents, but a lot of cases show that a word with many meanings makes the requirements definition ambiguous. To optimize the requirements definition process, we present a model in which the trust management process is integrated into the requirements definition process, in order to minimize the gap between requirements caused due to a lack or discrepancy in communication and to use a negotiation method for solving problems of this kind. We discuss that building trust between the stakeholders in the requirements definition process is effective to optimize the requirements definition, which has been produced by the special characteristics of Japanese firms in the information system development, and we also describe the necessity and effectiveness of the information system in Japan. In conclusion, it can be said that trust management using "trust" between the stakeholders to overcome the difficulty of requirements definition is higher in Japanese firms than in the U.S firms.**

Keywords- requirements definition; establishment of mutual trust; social uncertainty; the adaptive processes; special characteristics of Japanese firms.

I. INTRODUCTION

A. Background of Study

The role of the information system has increased in importance, such that the information system is now indispensable for companies to carry out work. Further, the information system has been considered as not only a useful product in companies, but also a significant infrastructure in the whole society. As the social mechanism has grown more sophisticated, the information system infrastructure has also continued to become more complicated and the difficulty of introducing it has increased steadily.

Kiritani has already advocated a model of trust management in the requirements definition that is a key success factor in the information system construction and it is optimized through the trust relationship between the stakeholders [1]. In addition, the author believes that the trust management is fully effective because the information technology development environment and method are peculiar to the Japanese firms.

B. Purpose of Study

The purpose of this study is to probe the following two items in order to verify the effectiveness of trust management in the requirements definition.

- To clarify the difference in the requirements definition between Japanese and U.S. firms when constructing the information technology system.
- To make sure that the trust management in the requirements definition is necessary and effective due to the special characteristics of the Japanese firms.

II. DIFFICULTY OF EXECUTING REQUIREMENTS DEFINITION IN JAPANESE FIRMS

A. Actual Condition of Information System Construction

The information technology system construction (ITSC) project that regards software creation as the main work is difficult to meet the needs and demands of customers through quality, cost and delivery. Approximately 70% of all ITSC projects had a problem with the quality, cost or delivery [1]. In addition, the survey results indicate that approximately 18% of all constructed IT systems have not actually been used, even after these projects were completed, or other projects were interrupted before their completion [2].

The reason ITSC is difficult to perform is considered to be the characteristics of the IT system itself: "the system is a complex aggregation of subsystems that have their own role and function relationally one another" or "the needs of customers in the IT system are always changing and a request for change is often submitted in the middle of the ITSC project" [3]. As the background of the ITSC project that ends in failure, some researchers pointed out that "this project is carried out with unclear customer needs"; "the customer needs are not easily settled"; "the grounds for estimation are ambiguous"; and "the system development plan is carelessly made". Other researchers also pointed out

 ISBN: 978-1-61208-391-9

that "the original documentation is insufficient" and "the project management is not conducted" [4].

The requirements definition that belongs to the upper process in the ITSC plays a role in compiling vague requirements from stakeholders into drawings and documents including the designable content for system installation from the viewpoint of technology, operation and expense. In other words, the requirements definition is an essential process to access the system development life cycle (SDLC), which is related to the following processes (from design process to operation/maintenance process) in the ITSC, and to affect decisively the quality, cost and delivery of the ITSC project. It seems that a request for change the abovementioned is submitted because the requirements are decided unclearly in the requirements definition and there is a gap between such requirements and the ones actually needed. Many causes of project failure arise from the method of executing the requirements definition and its results. Especially, an occurrence of trouble caused by the existence of tacit requirements and ambiguous consensus building that are not expressed clearly leads directly to the project failure under the limited man-hours and time for the requirements definition process.

A lot of investigations and researches have reported the relationship between the final success or failure of system development project and the requirements definition. For example, B. S. Blanchard [6] has reported that decision making at an early stage in the system development life cycle determines 70% to 80% of life cycle cost (LCC).

B. W. Boehm [7] has verified that the cost of modification or alteration (rework) traced to the first project management process increases as the project progresses on the basis of statistical data analysis in the system development project. K. E. Wiegers [5] has reported that the "rework" cost of the constructed system occupies 30% to 50% of all system development cost and the rework cost caused by errors in the requirements definition accounts for 70% to 85% of all the rework costs. In addition, other researchers have tried to explain the relationship between the quality of findings from software requirements specifications and the final success or failure of system development project [8]. The requirements definition is an important work process in the ITSC and the gap between requirements is pointed out as the most important cause of failure in the failed information system construction projects [9].

As a result, the quality of requirements definition leads directly to the final success or failure of system development project.

B. Difficulty of executing requirements definition causing ambiguity

It is evident that the requirements definition for the ITSC is carried out through communication between human beings, which accounts for a very large percentage of the requirements definition [10][11]. The ITSC including many distributed cooperative projects increases its difficulty level especially due to communication problems [12].

Language is used as the general method of compiling the requirements definition into drawings and documents, but, in a lot of cases, a word with many meanings makes the requirements definition ambiguous. A study report has pointed out that the effectiveness of communication using language is low [4]. Even if the same word is used, its meanings may be different between stakeholders because of their own background and interests.

Since the information technology system is usually equipped with massive functions, all functions of this system to be constructed are generally defined. However, it is necessary to design even the parts or components with undefined requirements to install them in the following processes and also, these undefined requirements are treated as tacit ones. The tacit requirements are often admitted in accordance with each stakeholder's "common sense", which brings a major cause of ambiguity of requirements definition (a gap between requirements).

The stakeholders who have really different backgrounds socially and economically take part in the requirements definition. Interests exist between the stakeholders, individual requirements of the stakeholders intertwine with their acknowledgement and thoughts on the project, and the requirements definition finishes without enabling the stakeholders to put everything in common. In this case, the stakeholders agree with one another in the style of "scrambling for the pie". Once trouble occurs, however, the stakeholders consider excessively their interests and requirements and negotiate ineffectively with one another for a solution to the problem, so that it will cause a serious problem that affects the quality, cost and delivery of the ITSC project.

III. PRESENTATION OF HYPOTHESIS AND RESEARCH QUESTIONS

A. Hypothesis

Because the requirements definition process indicates a lack of the necessary information on the partner's intention, it can be said that the social uncertainty exists here [14]. In this condition, a trust relationship between the stakeholders plays a role as lubricant for the social exchange relationships [14]. Therefore, it is believed that an improvement in the communication quality based on trust can minimize the gap between requirements in the requirements definition process.

The development of trust relationships changes the negotiation style between stakeholders from "scrambling for the pie" to "solving a problem" [13], so that it seems that the latter style improves the effectiveness of negotiation to solve the problem that affects the quality, cost and delivery of the ITSC project. The information technology system construction in Japanese firms presents unusual environment and high complexity, when it is viewed from the U.S. firms point of view. The main causes are the difference in the purpose and measures of system construction between Japanese and U.S. firms, the old business custom and the characteristics of a nation. Therefore, the development of trust relationships performs validly to solve problems usually caused in the requirements definition process especially in Japan.

 ISBN: 978-1-61208-391-9

We think that trust is important from an example of the definition of requirements to adaptive requirements specifications of the information system development in Japan and we made a model. Furthermore, we compared the circumstances with the ones in the United States.

B. *Research Questions*

- To clarify the difference in the requirements definition between Japanese and U.S. firms when constructing the information technology system.
- To make sure that the trust management in the requirements definition is necessary and effective due to the special characteristics of the Japanese firms.

IV. REVIEW OF PREVIOUS STUDIES

A. *Study on Optimization of Requirements Definition (Study on Trust Management in Requirements Definition)*

Kiritani designed, as a model of trust management in the requirements definition, its improved model that the effective optimization of the requirements definition can be expected to enhance communication and negotiation that are the great two factors in the requirements definition process through the development of trust relationships between the stakeholders [1].

1) Communication efficiency is improved by the development of trust relationships, minimizing the gap between requirements

The improvement of efficiency and accuracy of communication, which depends on the stakeholders, or human beings, through the establishment of mutual trust relationship minimizes the gap between requirements caused by their knowledge and recognition. With the development of trust relationships, an effect of reducing gap recognition can be expected for emotional requirements/needs (especially dispensable requirements/needs).

2) Realization of effective negotiation process with the development of trust relationships

Because the ITSC is limited by the quality, cost and delivery, it is difficult for the stakeholders to agree with one another after all requirements that are defined in the requirements definition process to remove completely the gap recognition between the stakeholders. A technique to allow the effective negotiation process to be performed in order to cope with the problem present in the following processes is required by mutual agreement on the assumption that there is the representative requirements gap including the tacit requirements.

Figure 1 shows an effect on the basic model of requirements definition when the mutual trust relationship is established using the trust management.

B. *Previous Studies on Comparison of Information*

Some information technology systems in the Japanese and U.S. firms are explained in the following investigations and studies: “Comparative Analysis of Japanese and U.S. Firms on IT and Management” [15], “International Comparison of IT Strategy and Company Performance between Japan, U.S. and South Korea” [16], “Survey of IT Usage in Corporate Management” [17], “Comparative Analysis of IT Management and Productivity between Japanese and U.S. Firms” [18] and the report on the system development in Japan compiled through interviews with persons having technical knowledge about the situation of system construction in both Japan and U.S. [19]. By reference to these investigations and studies, we clarify the difference in the IT system development between Japanese and U.S. firms.

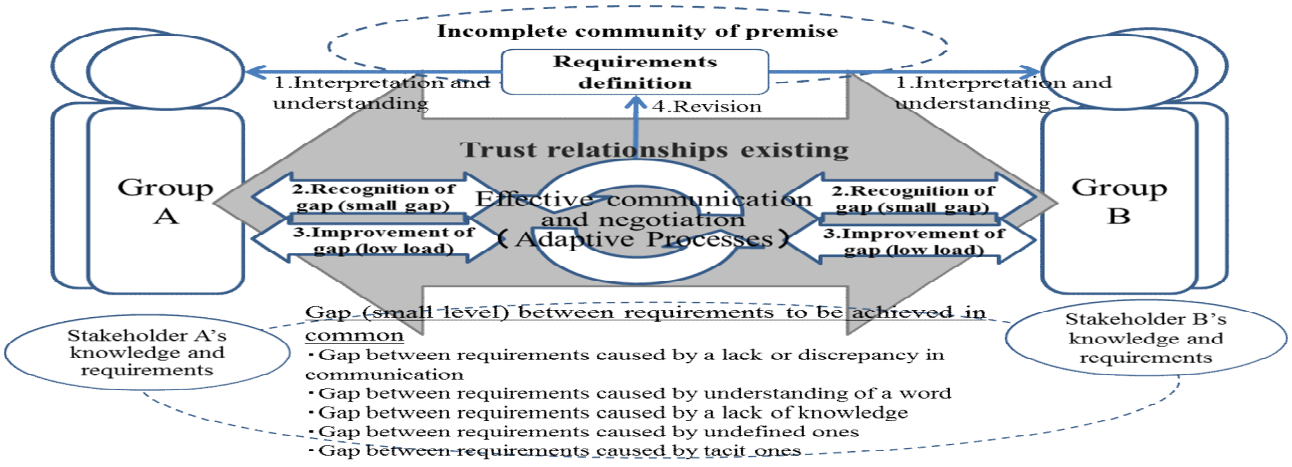


Figure 1. Basic model of requirements definition improved by trust management

1) Recognition of importance of IT investment and purpose of IT investment

There is a significant deference in the attitude toward “IT/information system investment” between Japanese and U.S. firms. While 75% of the U.S. firms consider this

 ISBN: 978-1-61208-391-9

investment as "a matter of great importance", only 16% of the Japanese firms keep the same policy. In terms of IT expectation, the Japanese firms rank "optimization of work using IT system and cost reduction (48.2%)" at the top of their list of priorities, but the U.S. firms rank first "strengthening development of products and services (41.0%)" and second "business model evolution" [15].

2) Role of Chief Information Officer(CIO)

The U.S. firms "employing an expert IT manager" are 38%, i.e., the highest percentage. However, most of the IT managers of the Japanese firms also hold the manager post of a different department (38%). As to the CEO's business career in the Japanese firms, many IT managers come from the in-house departments such as "corporate planning department", "general affairs/finance departments" and "information processing department" and are less than 30% under the equality of opportunity. On the other hand, 35% of the IT managers of the U.S. firms are scouted from outside the company and are extremely higher than those of the Japanese firms, which indicates that the U.S. firms appoint remarkably many outside specialists [16].

3) Software investment by type

The packaged software is less than 10% in Japan, but it is approx. 30% in the U.S. Most of software investment in Japan is made in order software. The feature of the Japanese software industry is the multi-step system that a major software company receives a large-scale system development order and divides it into small lots to place orders with the software houses as subcontractors [17].

4) Many Japanese firms dump everything on vendors and many U.S. firms performing in-house production

A lot of the U.S. firms have not performed IT system outsourcing activities by industrial field. As to the whole enterprise systems including the financial accounting, 58% of the U.S. firms have responded "no outsourcing activities", and as to the supply chain management (SCM) and sales promotion support, 53% of them have not requested outsourcing services. Although there is little difference in the ordering process between the Japanese firms, many of them have used outsourcing services.

Approx. 40% of the Japanese firms usually "make an order after exchanging contracts through previous negotiations between the outsourcing companies and their own companies" in all cases of "the whole enterprise systems including the financial accounting", "the systems by business department" and "the existing systems update", which explains a characteristic of the Japanese firms that carry out previous negotiations with the outsourcing companies. A lot of Japanese firms have responded that the IT system outsourcing companies are "needed to reduce costs". On the contrary, a few U.S. firms require the IT system outsourcing companies in order to reduce costs, but many U.S. firms expect these outsourcing companies to fill the role of a "technical adviser" [16].

C. Difference in Information System Development Method between Japanese and U.S. Firms

The difference in information system development method between Japanese and U.S.firms, including the study on the system development in Japan compiled through interviews

TABLE Ⅰ. DIFFERENCE IN IT SYSTEM DEVELOPMENT METHOD BETWEEN JAPAN AND U.S. FIRMS

Comparison items	Japanese firms	U.S. firms
Recognition of importance of IT investment	"Great importance": 16%	"Great importance": 75%
Purpose of IT investment	Top rank: "Optimization of in-house work and reduction of working time (35%)"	Top rank: "Speedup and optimization of product and service supply (45%)"
Role of CIO	Many IT managers also hold the manager post of a different department and come from the in-house departments.	Expert IT managers. Many of them are scouted from outside the company and are remarkably appointed as the outside specialists.
Software type	Software development order: approx. 80%	Packaged software: 30% and in-house software development: 35%
Management	・Users dump software development on vendors. ・The system department enters between vendors and user support department in order to lead and adjust the project.	・The in-house system development department leads the project and the inside engineers manufacture products in the company. The vendor as a technical adviser takes part in the project.
	・The user support department has authority to decide main requirements such as performance and operability. ・It is extremely important to make the user support department participate in the project in order to reflect correctly the business needs to the system. ・Even as premises for packaged software, the system is customized to meet the present requirements.	・While taking into account the requirements of the user support department, the CIO or the system department considers and determines the system design and specifications in the top-down decision-making process.
Relationship between business and system	The company manufactures the system to automate the completely preset business process	Business process re-engineering (BPR) is set as a precondition. The company creates a new flow of better business process and manufactures the system to make it smooth.
Software development method	Waterfall (The company regards the document and plan as important and pushes forward the final process without changing the specifications of software once these are set out.)	・Agile (The company uses short timeboxes called an "iteration" to minimize risk.)
Preconditioned quality	The company seeks complete quality on release	The company increases quality after release
Engineers' attitude	・The company regards technical achievement of engineers as important (conservative). ・The engineers accept a change in the specifications of software based on actual decisions of others because they wish to avoid trouble rather than right or wrong.	・The company regards innovativeness of engineers as important (aggressive). ・Since the rules to change the requirements and pecifications of software are clearly described in the documents attached to the contract sheets, the engineers have common sense to obey the rules when those are changed.
Estimation	With unclear customer needs, the user support company requests the vendor to submit a rough estimation.	The user support company submits a rough estimation based on its business knowledge and survey.

 ISBN: 978-1-61208-391-9

with persons having technical knowledge about the situation of system construction in both Japan and U.S. [19], is described below by reference to the previous studies on information technology system and the survey results. See Table 1. Difference in IT system development method between Japanese and U.S. firms.

V. ASSESSMENT OF TRUST ITEMS IN REQUIREMENTS DEFINITION

A. *Difference in Stakeholder Composition in Requirements Definition*

According to the following information system development and management model diagram, the features of stakeholders in information system construction are as follows:

- In Japanese firms, the information system development and management model are constructed complicatedly and the interests of the stakeholders are intricately intertwined.
- In the U.S. firms, the information system development and management model shows that the interests of the stakeholders are basically matched.

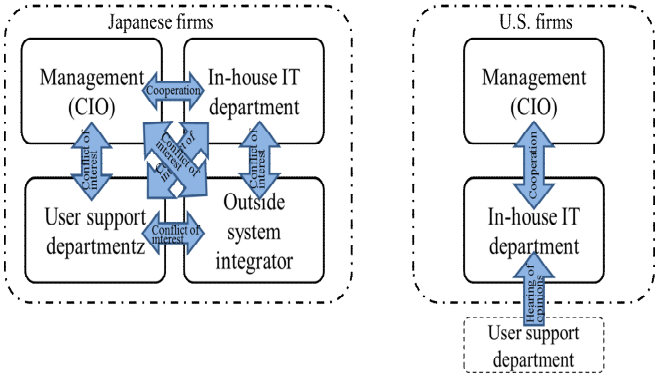


Figure 2. IT system development and management model in Japanese and U.S. firms

	Japanese firms	U.S. firms
Purpose of systematization	Top rank: "Optimization of in-house work and reduction of working time (35%)"	Top rank: "Speedup and optimization of product and service supply (45%)"
Relationship between business and system	The company manufactures the system to automate the completely preset business process	Business process re-engineering (BPR) is set as a precondition. The company creates a new flow of better business process and manufactures the system to make it smooth.
Recognition of requirements definition	Preconditions of removing ambiguity and preventing regression	Preconditions of allowing ambiguity
Software development method	Waterfall method	Agile method
Quality target at completion (on release)	No bugs	A few bugs are permitted, but they are worked out during actual operation.

Figure 3. Difference in recognition of requirements and quality between Japanese and U.S. firms

B. *Difference in System Development Method due to Recognition of Unclear Requirements Definition*

A lot of the Japanese firms manufacture the system to automate the completely preset business process for the purpose of systematization. However, the U.S. firms tend to create a new proper business process and manufacture the system to support it.

Therefore, there is a difference in recognition of unclear requirements definition between Japanese and U.S. firms. As a result, it seems that the Japanese firms often adopt the waterfall software development method that regression is not supposed, but the U.S. firms select the agile software development method that the improvement of software is postulated to repeat even after operation with the requirements not settled in the requirements definition process.

Because the outside system integrator as a consignee manufactures the system in Japan, this company may deliver the system at completion and aims to make a more perfect system, which affects the selection of software development method.

VI. DIFFERENCE IN INFORMATION SYSTEM DEVELOPMENT CONDITION BETWEEN JAPANESE AND U.S. FIRMS AND DISCUSSION OF TRUST MANAGEMENT IN REQUIREMENTS DEFINITION

A. *Difference in Requirements Definition Process between Japanese and U.S. Firms*

1) Information system development project in the Japanese firms showing the situation that "social uncertainty exists"

The information system development and management model in the Japanese firms indicates that the interests of the stakeholders are more intricately intertwined than that in the U.S. firms. This is because "the requirements definition needs information on the partner's intention, but is lacking in such information"; "it shows that the stakeholders have interests one another and social uncertainty escalates"; and "the partner's self-interest behavior makes you have a bad time". In other words, it can be said that the social uncertainty exists.

2) Information system development in the Japanese firms using the precondition and method of keeping consistent requirements

The Japanese firms select the information system development method to keep more consistent requirements as a precondition than the U.S. firms. As a result, the inconsistent requirements greatly affect the quality, cost and delivery. For this reason, it is necessary to optimize more the requirements definition process. It means that the measures used to optimize the requirements definition, such as the trust management, are extremely needed.

 ISBN: 978-1-61208-391-9

3) High quality required for the completion of information system in the Japanese firms

Since a lot of the Japanese firms order collectively the information system to an outside system integrator, the completion of information system leads to its delivery. Thus, a high quality level is needed for the information system. It can be also said that high accuracy is required for the requirements that are connected directly to the quality of information system.

B. Special Characteristics of Japanese Firms in Requirements Definition and Assessment of Trust Management

1) Special characteristics of the Japanese firms in comparison with the U.S. firms

The information system development and management model, including the requirements definition, in the Japanese firms can be extremely difficult because of the following:

- In the Japanese firms, the interests of the stakeholders are intricately intertwined in the requirements definition process, and it is more difficult to collect opinions and agree in the Japanese firms than the U.S. firms.
- The waterfall method that many Japanese firms use cannot remove an effect in the quality, cost and delivery caused by the change of requirements in the following processes. Like this, it is indispensable for determining the requirements in the requirements definition process, and it is more difficult to do so in the Japanese firms than the U.S. firms.

2) Special characteristics of the Japanese firms and necessity of trust management

In order to optimize the requirements definition process in the Japanese firms while taking into account their special characteristics in the requirements definition, trust management may be needed to optimize the requirements definition by increasing trust between the stakeholders.

VII. CONCLUSION

The interests of many stakeholders for the IT system construction in the Japanese firms are intricately intertwined, so that it can be clearly said that the IT system development project indicates “the situation that social uncertainty exists”.

The quality of information system is highly expected to improve in the Japanese firms and most of them use the construction method (Waterfall) with the precondition of keeping the consistent requirements, which causes finally troubles due to unclear customer needs.

On the other hand, the number of stakeholders in the U.S. firms is limited and a few conflicts of interests are considered between them. In other words, it can be said that “the social uncertainty does not exist”. As to the expectation of IT system quality to be improved, most of the U.S. firms use the construction method (Agile) with the precondition of keeping the inconsistent requirements and they can cope flexibly with the change of requirements.

In conclusion, it can be said that the trust management using “trust” between the stakeholders to overcome the difficulty of requirements definition is higher in the Japanese firms than in the U.S. firms;it is more effective and necessary to optimize the requirements definition in the information system development of Japan where a great effect is made by the change of requirements and the social uncertainty exists in the information system construction project.

REFERENCES

[1] Kiritani, K., Modeling Study on Trust Management in Requirements Definition, 2014

[2] Nikkei Computer, 2003 Survey in Information Technology, Vol. 17th of November, 2003, pp. 50-71

[3] The Standish Group International, 2004 Third Quarter Research Report, （http:// www。standishgroup.com）.

[4] Metzger.P. and Boddie, J. Managing a Programming Project, Third Edition. Prentice-Hall,New Jersey (1996).

[5] IPA SEC (Information-technology Promotion Agency, Japan, Software Engineering Center), Questionnaire Final Report by Software Development Expo & Conference, June 29 to July 1, 2005

[6] Blanchard, B.S., System Engineering Management. John Wiley & Sons, New York (1991).

[7] Boehm, B.W., Software Engineering Economics. Prentice-Hall, New Jersey (1981).

[8] Wiegers.K. E. Software Requirements, Microsoft Press, Washington (2003).

[9] Kamata, M. and Hosokawa, N., Findings from Software Requirements Specifications, Technical Reports of Information Processing Society of Japan, (Information System and the social environment), 2005, pp. 9-16

[10] Kiritani, K. Amitani, M. Ishizaka, H., Information System Construction Successful in Human Communication, Chuokeizai-Sha, 2011,pp.123-140,167-208

[11] Kiritani, K. Imamura, S. Kyougoku, T. Ishizaka, H., Textbook for Professional Requirements Definition,Chuokeizai-Sha, 2012, pp.1-42,56-80

[12] Kiritani, K. Ichikawa, H. Hirose, Y. Ymamamoto, Y., Actual Requirements Definition and Potentialities of Telework Establishment of Mutual Trust) ,Japan Telework Society, Proceedings of 15th Research Exhibition, 2013

[13] Yamagishi, T., Trust Construction – Evolutionary Game for Person’s Mind and Society, the University of Tokyo Press, 1998, pp. 13-15

[14] Sugita, K., Trust-Building Strategies in Negotiations, 2012

[15] Japan Electronics and Information Technology Industries Association, Comparative Analysis of Japanese and U.S. Firms on IT and Management, 2013

[16] The Research Institute of Economy, Trade and Industry, International Comparison of IT Strategy and Company Performance between Japan, U.S. and South Korea, 2007

[17] Economic Research Office of Information and Communications Policy Bureau of the Ministry of Internal Affairs and Communications, Survey of IT Usage in Corporate Management, 2005

[18] Kazuyuki Motohashi, Comparative Analysis of IT Management and Productivity between Japanese and U.S. Firms, Bank of Japan Working Paper Series No. 10-J-2, 2010

[19] Nikkei Systems, System Development in Japan Partly Seems Funny, 2012.

 ISBN: 978-1-61208-391-9

A Conceptual Framework for Guiding the Development of Feedback-Controlled Bulk Data Processing Systems

Martin Swientek
Paul Dowland

School of Computing and Mathematics
Plymouth University
Plymouth, UK
e-mail: {martin.swientek, p.dowland}@plymouth.ac.uk

Bernhard Humm
Udo Bleimann

Department of Computer Science
University of Applied Sciences Darmstadt
Darmstadt, Germany
e-mail: {bernhard.humm, udo.bleimann}@h-da.de

***Abstract*—The design, implementation and operation of an adaptive enterprise software system for bulk data processing differs from common approaches to implement enterprise systems. Different tasks and activities, different roles with different skills and different tools are needed to build and operate such a system. This paper introduces a conceptual framework that describes the development process of how to build an adaptive software for bulk data processing. It defines the needed roles and their skills, the necessary tasks and their relationship, artifacts that are created and required by different tasks, the tools that are needed to process the tasks and the processes, which describe the order of tasks.**

Keywords–adaptive middleware; software development process

I. INTRODUCTION

Enterprise Systems for bulk data processing are increasingly required to provide near-time processing of data to support new service offerings.

Traditionally, enterprise systems for bulk data processing are implemented as batch processing systems [1]. Batch processing delivers high throughput but cannot provide near-time processing of data, that is the end-to-end latency of such a system is high.

A lower end-to-end latency can be achieved by using single-event processing, for example by utilizing a message-oriented middleware for the integration of the services that form the enterprise system. While this approach is able to deliver near-time processing, it is hardly capable for bulk data processing due to the additional communication overhead for each processed message. Therefore, message-based processing is usually not considered for building a system for bulk data processing requiring high throughput [2].

The processing type is usually a fixed property of an enterprise system that is decided when the architecture of the system is designed, prior to implementing the system. This choice depends on the non-functional requirements of the system. These requirements are not fixed and can change over time.

Additionally, enterprise systems often need to handle load peaks that occur infrequently. When the system faces moderate load, a low end-to-end latency of the system is preferable. During the peak load, it is more important that the system can handle the load at all. A low end-to-end latency is not as important as an optimized maximum throughput in this situation.

For example, a billing system for a telecommunication carrier with moderate load over most of the time, but there are certain events with very high load such as New Year's Eve. Most of the time, a low end-to-end latency of the system is preferable when the system faces moderate load. During the peak load, it is more important that the system can handle the load at all. A low end-to-end latency is not as important as an optimized maximum throughput in this situation.

In [2], we have introduced the concept of a middleware that is able to adapt its processing type fluently between batch processing and single-event processing. By adjusting the data granularity at runtime, the system is able to minimize the end-to-end latency for different load scenarios.

The design, implementation and operation of such a system differs from common approaches to implement enterprise systems:

- There are specific activities or tasks needed to implement the feedback-control subsystem.
- There are roles needed with different skills.
- There are different tools needed to aid the design and development of such a system.

Developing software is a complex process, the quality of a software product depends on the people, the organization and procedures used to create and deliver it [3].

This paper introduces a conceptual framework to guide the design, implementation and operation of an adaptive system for bulk data processing. It defines views, roles, tasks and their dependencies, and processes to describe the necessary steps for design, implementation and operation of an adaptive system for bulk data processing.

Figure 1 shows an overview of the conceptual framework. It is organized among the phases plan, build and run. Each phase contains tasks, which are relevant for each phase:

 ISBN: 978-1-61208-391-9

- **Plan**
  The plan phase contains tasks relevant for the analysis and design of the system, such as the definition of the service interfaces, definition of the integration architecture and definition of performance tests.
- **Build**
  The build phase contains tasks relevant for the implementation of the system, such as the implementation of services, implementation of the integration layer and the implementation of the feedback-control subsystems.
- **Run**
  The run phase contains tasks relevant to the operation of the developed system, such as monitoring, setup and tuning.

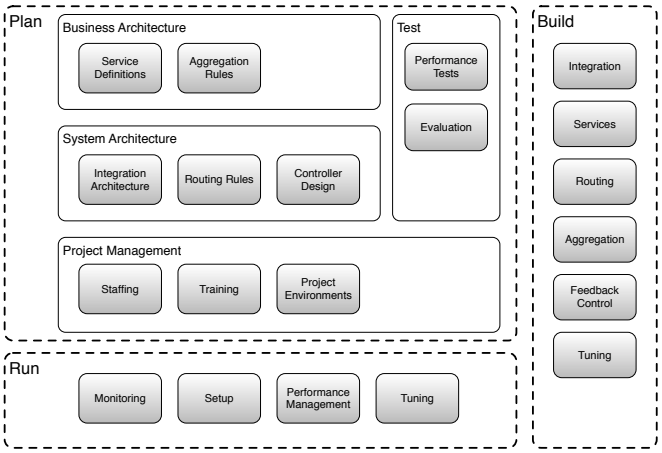


Figure 1. Overview of Conceptual Framework

The conceptual framework only describes concepts that are specific to the design and implementation of an Adaptive Middleware as described in the previous chapter. It does not describe common concepts for software development.

The remainder of this paper is organized as follows. Section II briefly introduces the concept of an adaptive middleware for bulk data processing. The conceptual framework is presented in Section III. Section IV gives an overview of other work related to this research. Finally, Section V concludes the paper and gives an outlook to the next steps of this research.

II. BACKGROUND

This section briefly introduces the concept of an adaptive middleware, which is able to adapt its processing type fluently between batch processing and single-event processing.

The middleware continuously monitors the load of the system and controls the message aggregation size. Depending on the current aggregation size, the middleware automatically chooses the appropriate service implementation and transport mechanism to further optimize the processing [2].

Figure 2 shows an overview of the adaptive middleware and its components.

The components of the middleware are based on the Enterprise Integration Patterns described by [4], as shown in Table I.

TABLE I
COMPONENTS OF THE ADAPTIVE MIDDLEWARE. WE ARE USING THE NOTATION DEFINED BY [4]

Symbol	Component	Description
	Message	A single message representing a business event.
	Message Aggregate	A set of messages aggregated by the Aggregator component.
	Queue	Storage component which stores messages using the FIFO principle.
	Aggregator	Stateful filter which stores correlated messages until a set of messages is complete and sends this set to the next processing stage in the messaging route.
	Router	Routes messages to the appropriate service endpoint.
	Service Endpoint	Represents a business service.

To control the level of message aggregation at runtime, the middleware uses a closed feedback loop with the following properties (see Figure 3):

- **Input (u):** Current aggregation size
- **Output (y):** Change of queue size measured between sampling intervals
- **Set point (r):** The change of queue size should be zero.

Preliminary tests show that the proposed middleware solution is viable and is able to optimize the end-to-end latency of a data processing system for different load scenarios [2].

III. CONCEPTUAL FRAMEWORK

The design, implementation and operation of a system based on the adaptive middleware introduced in Section II differs from common approaches to implement enterprise systems. We have therefore developed a conceptual framework to describe a development process how to build such a system.

A. Metamodel

The conceptual framework consists of the following entities, as shown in Figure 4:

- **Phase**
  Phases correspond to the different phases of a software development lifecycle, such as design, implementation and operations and contain the relevant tasks.
- **Task**
  Tasks represent the activities of the development process. A task

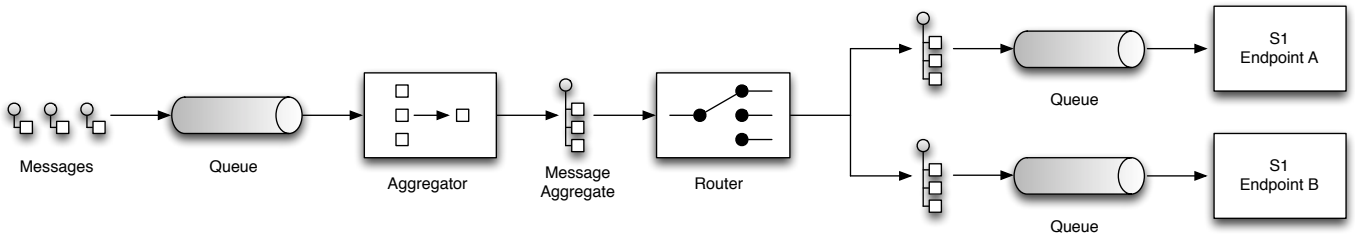


Figure 2. Overview of the adaptive middleware for bulk data processing [2]

- is contained in a phase
- is processed by a role
- produces and requires artifacts
- uses tools

- **Role**
  Roles represent types of actors with the needed skills to process specific tasks.
- **Artifact**
  An artifact represents the result of a tasks. Additionally, an artifact is a requirement of a tasks.
- **Tool**
  A tool is used by a tasks to produce its artifact.
- **Process**
  A process contains an ordered list of tasks that need to be processed in a certain order.

B. Roles

Roles represent the actors, which process tasks, that is, they describe *who* does something. The description of a role contains its responsibilities and needed skills. A role is not the same as a person, a single person can have multiple roles and change the role according to the context of the current task.

The conceptual framework defines the following roles:

- **Business Architect**
  The Business Architect is responsible for designing the business architecture of the system, including the definition of services and aggregation rules.
- **System Architect**
  The System Architect is responsible for designing the technical architecture of the system, including the integration and controller architecture.
- **Software Engineer**
  The Software Engineer is responsible for the implementation of the system, including the implementation and tuning of the feedback-control loop.

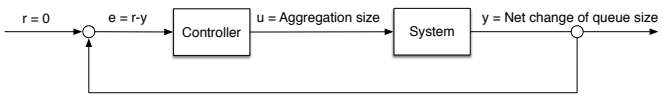


Figure 3. Feedback loop to control the aggregation size

- **Test Engineer**
  The Test Engineer is responsible for defining and performing the performance tests of the system.
- **Operations Engineer**
  The Operations Engineer is responsible for operating the system, including setup, deployment and monitoring.
- **Project Manager**
  The Project Manager is responsible for the project coordination, including the staffing and planing of the required environments.

A role is described by the following attributes:

- **Name**
  The name of the role.
- **Description**
  Description of the responsibilities of the role.
- **Tasks**
  The tasks the role is responsible to process.
- **Needed skills**
  The skills the role has to have in order to successfully process its tasks.

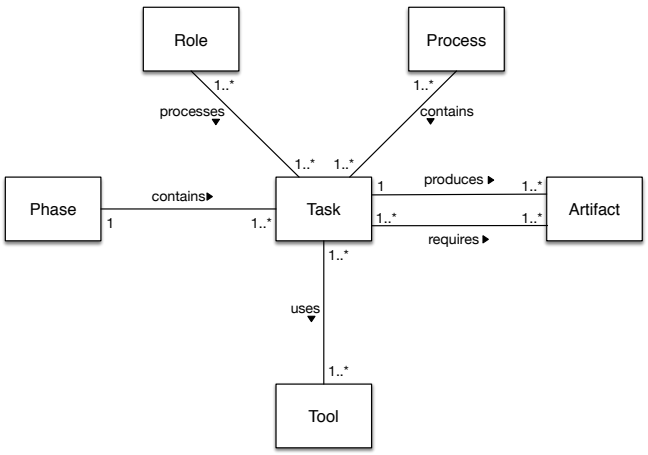


Figure 4. Metamodel

C. Tasks

Tasks are the main entities of the conceptual framework. A Tasks describes *what* should be done, *why* should it be done,

and *who* should do it. Additionally, it describes the required and produced artifacts, the tools that should be used to process the task and the expected challenges.

Tasks depend on each other, some tasks must be processed in a certain order. A task can have multiple subtasks.

The Conceptual Framework only describes tasks that are specific to the design and implementation of an Adaptive Middleware for Bulk Data Processing as described in [2]. It does not describe common tasks or activities that are needed for every software system.

Figure 5 shows an overview of the tasks grouped by the different phases of the Conceptual Framework.

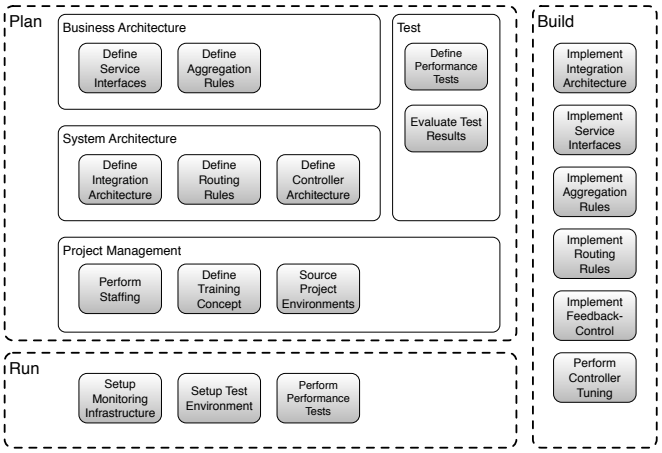


Figure 5. Overview of tasks

A Task is described by the following attributes:

- **Name**
  The name of the task.
- **What**
  Describes the content of the task.
- **Why**
  Describes the purpose of the task.
- **Who**
  Describes the roles, that are responsible for processing the task.
- **Input**
  The required artifacts of the task.
- **Output**
  The artifacts produced by the task.
- **Tools**
  The tools that are needed to process the task.
- **Challenges**
  Describes the expectable challenges when processing the task.

D. Processes

A process contains an ordered list of tasks that are concerned with the implementation of a certain feature of the software system. Processes are modeled using Unified Modelling Language (UML) activity diagrams. The conceptual framework describes the following processes:

- Implement Integration
- Implement Aggregation
- Implement Feedback-Control

1) Implement Integration: This process describes the necessary tasks to implement the integration layer and the integrated service interfaces, as shown in the UML activity diagram in Figure 6.

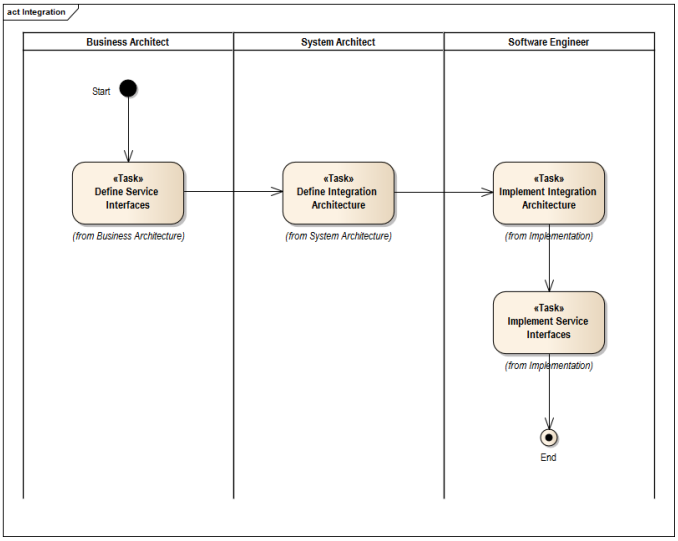


Figure 6. UML Activity Diagram: Implement Integration

2) Implement Aggregation: This process is concerned with the implementation of the message aggregation, as shown in the UML activity diagram in Figure 7.

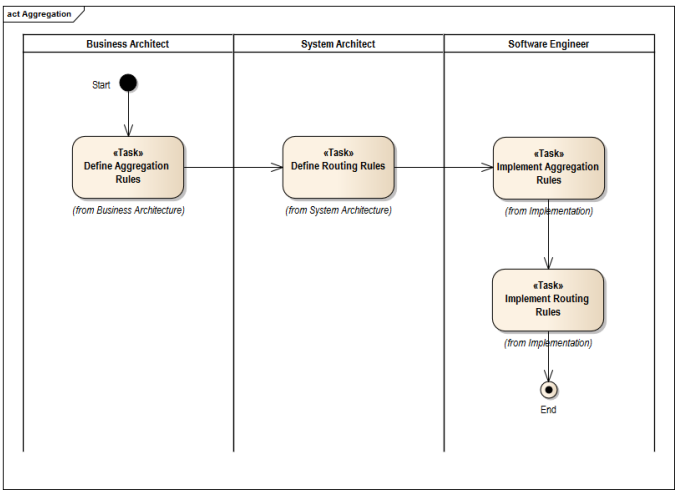


Figure 7. UML Activity Diagram: Implement Aggregation

3) Implement Feedback-Control: This process contains tasks that are concerned with the design, implementation and tuning of the feedback-control loop, as shown in Figure 8.

 ISBN: 978-1-61208-391-9

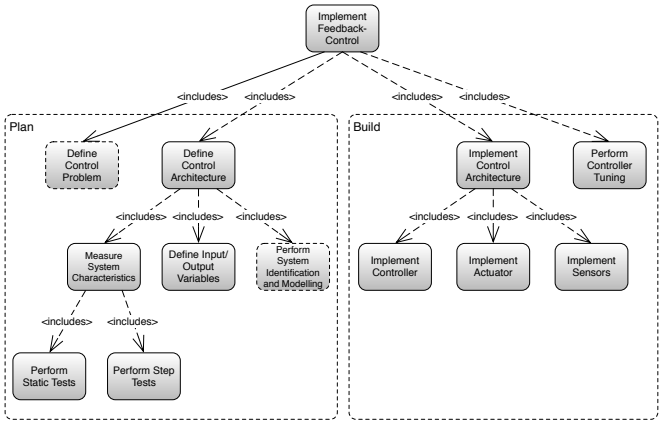


Figure 8. Tasks for implementing the feedback-control loop

There are two options for implementing the feedback-control loop:

- Using a system model for performing the controller tuning, as shown in the UML activity diagram in Figure 9.
- Without using a model, the control architecture needs to be implemented prior to the controller tuning, as shown in the UML activity diagram in Figure 10.

E. Artifacts

An artifact is a result of a task. It is an intermediate result, that is needed for development of the software, but not the software product itself. Additionally, it can also be prerequisite of another task.

The conceptual framework defines the following artifacts:

- **Performance Requirements**
  Defines the requirements regarding the performance of the system, such as required maximum throughput, required maximum latency or desired minimum latency. Defines the workload scenarios of the system.
- **Service Interface Definition**
  Defines the structure of input and output data. Does not include informations about the technical format, such as Extended Markup Language (XML) or JavaScript Object Notation (JSON), and the integration style, such SOAP or Representional State Transfer (REST).
- **Aggregation Rules**
  Defines how events should be correlated with each other by the Aggregator.
- **Integration Architecture**
  Defines the technical integration of the business services, including Middleware technology or product, transports, such as Java Messaging Service (JMS), SOAP or File Transfer Protocol (FTP), Technical format of the input and output data, such as XML or JSON, Comma Separated Values (CSV) or binary formats.

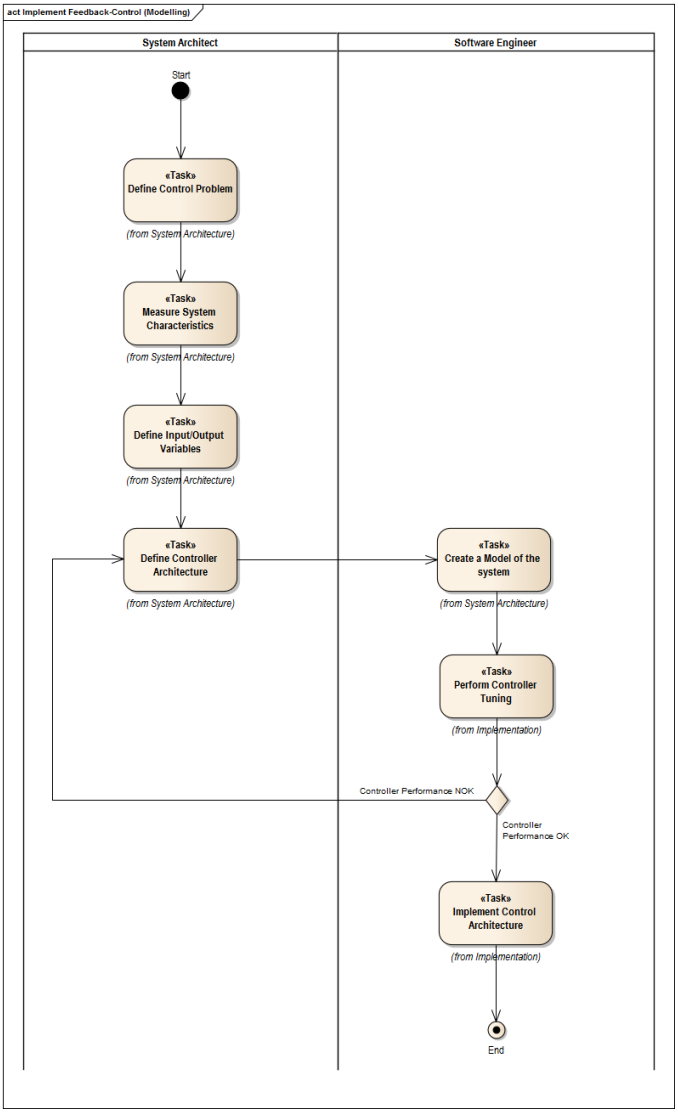


Figure 9. UML Activity Diagram: Implement Feedback-Control Loop using a model

- **Routing Rules**
  Defines which service endpoint should be called by the Router for a given aggregation size.
- **System Model**
  The system model is used to build a simulation of the system which can be used for implementing the controller.
- **Controller Configuration**
  The controller configuration specifies the parameter of the Controller.
- **Training Concept**
  Defines the training concept, including the audience, the content and the type of training. Additionally it contains a time-plan, learning modules and needed facilities to conduct the training.
- **Staffing Plan**
  Defines the required team members and their utilization over the project time (staffing curve), the required

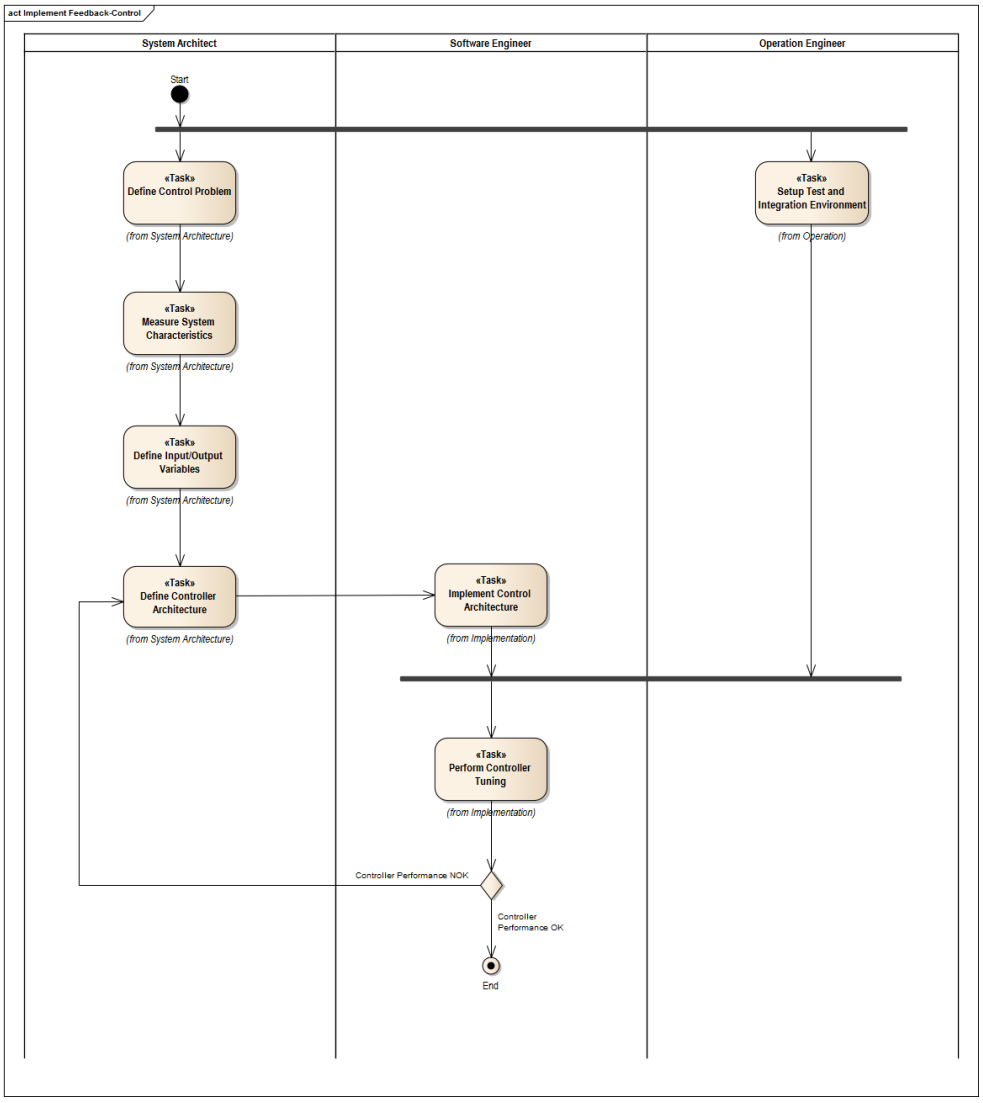


Figure 10. UML Activity Diagram: Implement Feedback-Control Loop without using a model

roles and their assignment to team members and a skill matrix that shows the required skills and the knowledge of each team member.

An artifact is described by the following attributes:

- **Name**
  The name of the artifact.
- **Description**
  A description of the artifact.
- **Task**
  The task that produces the artifact.
- **Role**
  The role that is responsible for producing the artifact.

IV. Related Work

This section discusses work related to the conceptual framework presented in this paper. It introduces the terms *Software Process* and *Software Process Modelling* and discusses approaches to model the software process using UML.

A. Software Process

"The software process is a partially ordered set of activities undertaken to manage, develop and maintain software systems." [5]

McChesney [6] describes the software process as "collection of policies, procedures, and steps undertaken in the transformation of an expressed need for a software product into a software product to meet that need.".

Another similar definition comes from Fugetta [3]. He defines the software process as the "coherent set of policies, organizational structures, technologies, procedures, and artifacts that are needed to conceive, develop, deploy, and maintain a software product."

It is necessary to differentiate between the terms software process and software lifecycle. A software lifecycle describes the states through which the software passes from the start of the development until the operation and finally the retirement. [7] Examples of software lifecycle models are the waterfall model [8] or the spiral model [9].

 ISBN: 978-1-61208-391-9

B. Software Process Modelling

Software process modelling describes the creation of software development models [5]. A software process model is "an abstract representation of a process architecture, process design or process definition, where each of these describe, at various levels of detail, an organization of process elements of either a completed, current or proposed software process" [10].

Process models are described using Process Modelling Languages (PMLs). A PML is defined in terms of a notation, a syntax and semantics, often suitable for computational processing [11].

Typical elements of PMLs are (see for example [12] [5] [3] [13]):

- Agent or Actor
- Role
- Activity
- Artifact or Product
- Tools

Process Models commonly use different perspectives to describe the software process [13]:

- Functional: what activities are being performed
- Behavioral: In which order (when) are activities performed
- Organizational: where and by whom is an activity performed
- Informational: the entities produced by the process

Examples of software process models include the IEEE and ISO standards IEEE 1974-1991, ISO/IEC 12207 and the Rational Unified Process (RUP).

C. Software Process Modelling using UML

UML is commonly used for modelling software processes.

UML for Software Process Modelling (UML4SPM) is an UML-based metamodel for software process modelling [11] [14]. It takes advantages of the expressiveness of UML 2.0 by extending a subset of its elements suitable for process modelling. UML4SPM contains two packages. The process structure package, which contains the set of primary process elements and the foundation package, which contains the subset of UML 2.0 concepts extended by this process elements to provide concepts and mechanisms for the coordination and execution of activities.

Software & System Process Modelling Metamodel (SPEM) 2.0 is a metamodel for modeling software development processes and a conceptual framework, which provides concepts for for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes [15]. It provides a clear separation between method content, for example deliverables and key roles, and workflows supporting different software lifecycle models. The SPEM 2.0 metamodel consists of seven main metamodel packages, with each package extending the package it depends on.

Both approaches, UML4SPM and SPEM 2.0 extend the UML 2.0 notation with additional elements, which does not allow the usage of standard UML tools.

[16] use UML 2.0 for modelling software processes at Siemens AG. According to the authors, the usage of standard UML 2.0 notation, which is supported by standard modelling tools, increases readability of processes for software developers since UML is also used for modelling the software itself. They describe four distinct process views, that are described by UML activity diagrams, class diagrams and use-case diagrams: process-oriented, activity-oriented, product-oriented, and role-oriented. The following UML diagram types are used by their approach:

The conceptual framework for feedback-controlled systems for bulk data processing presented in this chapter is based on the properties of the described approaches in this section for modelling the software development process. It uses standard UML use-case and activity diagrams for describing tasks and processes for the following reasons:

- **Understandability**
  Using standard UML 2.0 notation elements and diagrams facilitate the understanding of the conceptual framework since they are commonly used by software engineers for the design of the software system itself.
- **Tool support**
  Standard UML 2.0 notation elements and diagrams are supported by a wide range of modelling tools.

Standard metamodels for software process modelling such as SPEM 2.0 have not been used because they seemed to heavyweight for the intended purpose.

D. Software Processes for Adaptive Software Systems

It has been understood that software processes need to be reconceptualised to engineer self-adaptive software systems (see for example [17] [18] [19] [20]). Self-adaptive systems adjust their behavior automatically to respond to changes in their context and requirements. Activities that are traditionally done at development-time need to be shifted to run-time. Additionally, some activities that are previously performed by software engineers are now performed by the system itself. In a way, the role of the human software engineer is to some extend shifted from operational to strategic. The engineer implements the adaption mechanisms, the adaption itself is performed by system.

[20] extend the SPEM metamodel to specify which activities should be performed off-line and on-line and the dependencies between them. They distinguish between off-line activities, manual activities that are performed externally at development-time and on-line activities, that are performed internally at run-time, by the system itself, for example evolution and adaption activities performed by the adaption logic of the system. The authors argue, that on-line activities must be explicitly reflected in software process models, since they are not independent from off-line activities. In addition to on-line activities, on-line roles and work products also need to be

 ISBN: 978-1-61208-391-9

addressed by process models. To meet this requirements, they extend the SPEM metamodel with

- On-line and off-line stereotypes to define whether an activity should be performed on-line or off-line
- Dependencies to relate two or more arbitrary process elements
- Elements to describe the costs and benefits of performing an activity on-line in contrast to perform it off-line.

[21] describe a process methodology to support the development of context-aware adaptive applications. It consists of four different activities: *Explore*, *Integrate*, *Validate* and *Evolve*:

- **Exploration Phase**
  Exploits a feature library containing the implementation and corresponding requirements description.
- **Integration phase**
  Uses these features to produce a feature-diagram to describe the space of system changes, called variants.
- **Validation phase**
  Validates the variants by using context analysis and model checking.
- **Evolution phase**
  Reconfigures the system by switching to the new configuration.

[22] propose a conceptual model for self-adaptation which uses the ITIL Change Management process as a starting point. It consists of a reference process, activities, roles and responsibilities and artifacts. The reference process consists of two processes that interact iteratively, the adaption process and the evolution process:

- The inner *Adaption Process* relates to the feedback-loop of a single adaptable element of the system and is comprised of the activities *Sense*, *Trigger*, *Select Adaption Rules* and *Change*. All these activities are fully automated.
- The *Evolution Process* is executed for a single or multiple occurrences of the inner adaptive process. It consists of the activities *Aggregate Metrics*, *Analyze*, *Evolve Adaption Rules*, *Adjust and Synchronize*, and *Reflect*. Theses tasks might require human involvement.

The related work on process models for adaptive systems is focused on generic adaptation mechanisms to evolve and adapt a system, which are carried out at run-time. In contrast, the conceptual framework presented in this chapter is aimed to guide the design, development and operation of a specific system, that is, an adaptive system for bulk data processing, which provides a specific adaptation mechanism, that is, the adaption of the aggregation size at run-time depending on the current load of the system.

V. Conclusion

In this paper, we have presented a conceptual framework to guide the design, implementation and operation of an enterprise system that implements the adaptive middleware for bulk data processing as described in [2].

The conceptual framework consists of the entities phases, roles, tasks, artifacts and tools. It describes:

- The needed roles and their skills for the design, implementation and operation.
- The necessary tasks and their relationships for the design, implementation and operation.
- The artifacts that are created and required by the different tasks.
- The tools that are needed to process the different tasks.
- The processes that describe the order of tasks to implement a certain feature of the software system.

It should be noted that software processes are not fixed during their lifetime, they need to be continuously improved. [3] The conceptual model can therefore be tailored to specific projects requirements, it does not have to be followed strictly.

The next step of this research is the evaluation of the conceptual framework by using quantitative research methods, such as expert interviews and its application in real-life projects.

References

[1] J. Fleck, “A distributed near real-time billing environment,” in Telecommunications Information Networking Architecture Conference Proceedings, 1999. TINA ’99, 1999, pp. 142–148.

[2] M. Swientek, B. Humm, U. Bleimann, and P. Dowland, “An Adaptive Middleware for Near-Time Processing of Bulk Data,” in ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications, Venice, Italy, May 2014, pp. 37–41.

[3] A. Fuggetta, “Software process: a roadmap.” ICSE - Future of SE Track, 2000, pp. 25–34.

[4] G. Hohpe and B. Woolf, Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[5] S. T. Acuña and X. Ferré, “Software process modelling.” in ISAS-SCI (1), 2001, pp. 237–242.

[6] I. McChesney, “Toward a classification scheme for software process modelling approaches,” Information and Software Technology, vol. 37, no. 7, 1995, pp. 363–374.

[7] S. T. Acuña and X. Ferre, “The software process: Modelling, evaluation and improvement,” Handbook of Software Engineering and Knowledge Engineering, vol. 1, 2001, pp. 193–237.

[8] W. W. Royce, “Managing the Development of Large Software Systems: Concepts and Techniques.” ICSE, 1987, pp. 328–339.

[9] B. W. Boehm, “A Spiral Model of Software Development and Enhancement.” IEEE Computer (), vol. 21, no. 5, 1988, pp. 61–72.

[10] P. Feiler and W. Humphrey, “Software process development and enactment: concepts and definitions,” in Software Process, 1993. Continuous Software Process Improvement, Second International Conference on the, Feb 1993, pp. 28–40.

[11] R. Bendraou, M.-P. Gervais, and X. Blanc, “UML4SPM: A UML2.0-Based Metamodel for Software Process Modelling,” in Model Driven Engineering Languages and Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 17–38.

 ISBN: 978-1-61208-391-9

[12] K. Benali and J. C. Derniame, "Software processes modeling: What, who, and when," in Software Process Technology. Berlin/Heidelberg: Springer Berlin Heidelberg, Jan. 1992, pp. 21–25.

[13] B. Curtis, M. I. Kellner, and J. Over, "Process modeling," Communications of the ACM, vol. 35, no. 9, Sep. 1992, pp. 75–90.

[14] R. Bendraou, M.-P. Gervais, and X. Blanc, "UML4SPM: An Executable Software Process Modeling Language Providing High-Level Abstractions," in Enterprise Distributed Object Computing Conference, 2006. EDOC '06. 10th IEEE International, Oct 2006, pp. 297–306.

[15] OMG, "Software Process Engineering Metamodel SPEM 2.0," Object Management Group, Technical Report ptc/08-04-01, 2008.

[16] S. Dietrich, P. Killisperger, T. Stückl, N. Weber, T. Hartmann, and E.-M. Kern, "Using uml 2.0 for modelling software processes at siemens ag," in Information Systems Development, R. Pooley, J. Coady, C. Schneider, H. Linger, C. Barry, and M. Lang, Eds. Springer New York, 2013, pp. 561–572.

[17] G. Blair, N. Bencomo, and R. France, "Models@ run.time," Computer, vol. 42, no. 10, Oct 2009, pp. 22–27.

[18] P. Inverardi and M. Tivoli, "The Future of Software: Adaptation and Dependability." ISSSE, vol. 5413, no. Chapter 1, 2008, pp. 1–31.

[19] R. De Lemos, H. Giese, H. A. Müller, and M. Shaw, "Software engineering for self-adaptive systems: A second research roadmap," Software Engineering for ..., 2013.

[20] J. Andersson, L. Baresi, N. Bencomo, R. de Lemos, A. Gorla, P. Inverardi, and T. Vogel, "Software engineering processes for self-adaptive systems," in Software Engineering for Self-Adaptive Systems II, ser. Lecture Notes in Computer Science, R. de Lemos, H. Giese, H. Müller, and M. Shaw, Eds. Springer Berlin Heidelberg, 2013, vol. 7475, pp. 51–75. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35813-5_3

[21] P. Inverardi and M. Mori, "A Software Lifecycle Process to Support Consistent Evolutions." Software Engineering for Self-Adaptive Systems, vol. 7475, no. Chapter 10, 2010, pp. 239–264.

[22] C. Gacek, H. Giese, and E. Hadar, "Friends or foes?: a conceptual analysis of self-adaptation and it change management." SEAMS, 2008, pp. 121–128.

 ISBN: 978-1-61208-391-9

A Low Cost Adaptive Wireless Sensor Network for Accelerometer Data Collection

Ying Li, Peter Gould
Multiple Access Communications Limited
Southampton, UK
Email: ying.li@macltd.com, peter.gould@macltd.com

***Abstract*—This paper presents an accelerometer data collection system implemented with low cost off-the-shelf wireless sensor nodes. The design is focused on addressing some practical issues including effective sensor data offloading schemes suitable for different usage scenarios. Making the system highly configurable and adaptive in terms of selecting appropriate triggers for starting/stopping data collection and appropriate data offloading schemes was also a key design focus. The system has been developed for carrying out field tests, which involved attaching some sensor nodes to railway sleepers and gathering raw accelerometer data. The initial field test on an operational rail track proved the configurability and adaptability of our wireless sensor network system.**

Keywords-wireless sensor; sensor network; accelerometer.

I. INTRODUCTION

In recent years, there has been increasing interest in studying and developing wireless sensor network (WSN) systems that use accelerometers for monitoring problems relating to railway tracks or trains. [1] proposed a method of using accelerometers to detect arriving trains in order to warn maintenance personnel working on tracks. The authors of [2] and [3] investigated the detection and classification of train events by analyzing acceleration sensor data. [4] reported that by processing accelerometer readings measured at different locations of a train it is possible to distinguish between the vibration due to the train itself and the vibration due to deformation of the track. A prototype design of a WSN for monitoring a railway bridge is presented in [5]. It is evident from these studies that accelerometer sensor data is likely to play an important part in the application of the WSN technology to the railway sector. However, studies and research reported in the literature so far are mostly proposals, models and prototypes. What is needed is a robust WSN that could be used to make in-situ measurements in a reliable and flexible way.

Innovate UK has co-funded a two-year project called Smart Green Railway Sleepers (SGRS), which started in January 2014. One key aspect of this new project is to design, develop and pilot a wireless sensor-enabled tag and track system for use within railway sleepers (or railroad ties). Embedding sensors into railway sleepers opens a host of potential ways to improve the railway maintenance and sleeper recycling approaches. However, there is a range of challenging issues that need to be addressed before designing and embedding an optimal WSN system into railway sleepers becomes feasible, as highlighted in [6]. This paper reports on our experience of designing such an adaptive WSN system and provides an insight into practical issues encountered in collecting raw accelerometer data, along with approaches and methods used to address these issues.

The rest of this paper is organized as follows. Section II presents the main system requirements and a design overview. Section III details the challenging issues encountered in our design. Section IV presents our approaches and solutions. Section V draws conclusions to this paper.

II. SYSTEM REQUIREMENTS AND DESIGN OVERVIEW

Our aim is to design a low cost WSN for collecting raw accelerometer readings from sensors attached to railway sleepers while a train passes. The WSN should include a data sink node, a set of sensor nodes and optionally one or more relay nodes. The data sink node is connected to a laptop where an application controls the operation of the WSN system and also handles the collected measurement data, e.g., data visualization. One or more relay nodes are required in case the data sink node is located outside the radio range of the sensor nodes attached to railway sleepers. These requirements can be met by existing low cost off-the-shelf development boards, such as the CC2530ZNP-Mini kit from Texas Instruments [7].

The CC2530ZNP-Mini node includes two processors: a CC2530 system-on-chip running ZigBee Network Processor (ZNP) firmware and a MSP430F2274 microcontroller running application software, which controls the operation of the ZNP. Each CC2530ZNP-Mini node includes a 3-axis accelerometer that can be configured to sense acceleration in the range of ±2g or ±8g at a sampling rate of 10Hz, 40Hz, 100Hz or 400Hz. A CC2530ZNP-Mini node can be configured as a ZigBee coordinator, a ZigBee router or a ZigBee end device. We have used a coordinator node as a data sink node and end devices as sensor nodes.

An end device is designed to support the following simple operational states.

- ***Network Discovery***: At power on or wake up from sleep, a node searches for its network. If successful, it configures the accelerometer and enters the Wait for Event state. Otherwise, it enters the Sleep state.
- ***Sleep***: The Sleep duration is user configurable. At the expiration of the sleep time, an end device enters the Network Discovery state again.
- ***Wait For Event***: In this state, the MSP430F2274 stays in a low power mode until one of the following

 ISBN: 978-1-61208-391-9

three events occurs: 1) a trigger event for starting sensor data measurement; 2) the reception of a control message from the coordinator; 3) a detection of the network being lost. At the reception of a control message, the end device processes the message, e.g., changing motion detection threshold level, and remains in the current state. At the detection of the network being lost, the node enters the Sleep state. At the detection of a trigger event, the node enters the Data Measurement state.

- ***Data Measurement***: In this state, an end device measures accelerometer values and stores the measurement data locally. After a user configurable number of data samples have been collected, the end device enters the Data Offload state.
- ***Data Offload***: In this state, the sensor data collected by an end device is offloaded to the data sink node. Once the data has been offloaded, the end device goes back to the Wait For Event state.

The transitions between the above states are illustrated in Figure 1. Note that, when no network is present, an end device will enter the Sleep state in order to limit the power consumption of the end node.

The number of data samples collected and stored locally in a sensor node before being offloaded takes into consideration the memory available on the hardware platform and the usefulness of the measurement data. Among the available 32 KB FLASH space in a sensor node, 12 KB is reserved for buffering measurement data. This would give a maximum of 10 seconds worth of accelerometer readings at a sample rate of 400 Hz, which is sufficient for a useful fast Fourier transform (FFT) analysis.

To initiate a measurement session, the coordinator is plugged into a laptop via a USB port and forms a network for other nodes in its radio range to join. It has two main functions: 1) taking control and configuration commands from the application running on the laptop and transmitting them to end devices; 2) receiving accelerometer measurement data from end devices and passing them to the application running on the laptop where measurement data can be displayed graphically.

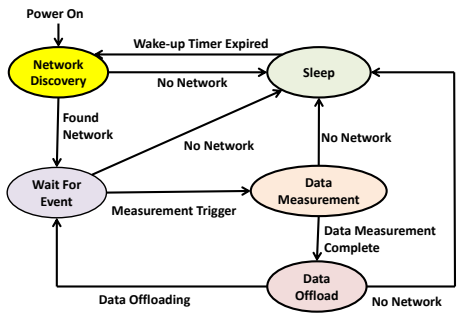


Figure 1. End device state transition diagram.

The data sink node is designed to pass each received packet to the laptop immediately for two reasons: firstly, there is not enough memory space for storing sensor data from more than one sensor node without adding external memory to the hardware platform; secondly, minimizing data processing at the data sink node reduces the time required for the measured sensor data to reach the laptop so that sensor data can be inspected in near real time.

III. CHALLENGING ISSUES

One of the challenging issues associated with this system is the design of data offloading schemes. Ideally, an end device starts to transmit collected measurement samples as soon as it enters the Data Offload state. The sooner a node offloads its collected data, the sooner it becomes ready for the next round of measurements. However, in an application where several nodes take sensor data measurements in a correlated manner, e.g., all sensor nodes start to take measurements as a train passes and want to offload their collected data as soon as possible, excessive packet collisions at the receiver of the data sink node would occur unless some sort of data offload scheduling scheme is deployed. Missing samples from a set of collected accelerometer readings could potentially lead to some distortions in data analysis results, rendering that whole data set useless. So, it is vitally important that a complete set of measurement samples reaches the data sink node reliably.

We could design a handshake process between the data sink node and each sensor node to schedule data offloading, e.g., messages to indicate which sensor node has data to offload and messages to dictate when and which sensor node should offload its data. This approach has at least two drawbacks. Firstly, this signalling would consume precious coding space in the MSP430F2274. Secondly, the signalling overhead increases when the number of sensor nodes increases and the channel condition gets poorer. For these reasons, we adopted an approach of making maximum use of what is available on the hardware platform.

The basic mechanism for reliable packet delivery in the presence of packet collisions and/or radio interference is packet acknowledgement and retransmissions. The ZNP offers two acknowledgement modes: medium access layer acknowledgement (MAC-ACK) mode and application support layer acknowledgement (APS-ACK) mode.

- ***In MAC-ACK mode***, the acknowledgement is from a neighbouring node. If there are multiple hops between a sensor node and the data sink node, receiving a positive MAC-ACK cannot be used as an indicator that a packet has reached the data sink node. The MAC-ACK mode is always on and cannot be disabled on this platform via the application programming interface (API). The two parameters governing the MAC-ACK mode operation are the maximum duration of waiting for an acknowledgement, T_{mac_wait}, and the maximum number of retries, N_{mac_retry}. These two parameters are also out of the control of the MSP430F2274 in this platform. However, the MSP430F2274 is informed of each packet transmission result: a positive result means that an acknowledgment to the packet has been received while a negative result

means that there is no acknowledgement for a packet after the packet has been transmitted N_{mac_retry} + 1 times and the T_{mac_wait} timer has expired after each transmission.

- ***In APS-ACK mode***, an acknowledgement is from the final destination node. Receiving an acknowledgement for a packet in APS-ACK mode is an indication that the packet has reached the ZNP of the data sink node. Unlike MAC-ACK mode, the APS-ACK mode can be enabled or disabled by the MSP430F2274 via the application programming interface (API). The maximum duration of waiting for an acknowledgement, T_{aps_wait}, and the maximum number of retries, N_{aps_retry}, for the APS-ACK mode operation are also under the control of the application processor.

One problem with MAC-ACK and APS-ACK is that an acknowledgement is sent by the ZNP. The ZNP and the MSP430F2274 processor in a node communicate via an internal serial peripheral interface (SPI). At the reception of a packet from a sensor node, the ZNP in the data sink node puts the packet into a buffer and notifies the MSP430F2274 that the packet is available via the SPI. ZigBee has an over-the-air data rate of 250 kb/s while the serial port data rate is configurable from 9.6kb/s to 115.2 kb/s. This means that while the MSP430F2274 processor is busy sending a received packet to the connected laptop, the next packet received and buffered by the ZNP may be over-written by subsequent packets, leading to the loss of one or more packets. To avoid losing packets in this way, one possible approach is for a sensor node to introduce a delay between receiving a positive acknowledgement for a packet and transmitting the next packet. If there is just a single sensor node offloading data, the sensor node could easily estimate the minimum delay. But in the case of multiple sensor nodes, estimating the required delay by a sensor node becomes difficult, especially when the number of sensor nodes offloading data varies. To ensure reliable end-to-end packet delivery, the application layer acknowledgement (APP-ACK) mode was introduced.

- ***In APP-ACK mode***, an acknowledgement is from the application layer of the final destination node. The two parameters, T_{app_wait} and N_{app_retry}, are used to denote the maximum duration for waiting for a positive acknowledgement and the maximum number of retries, respectively, at the application layer.

The three acknowledgement modes are illustrated in Figure 2. It is shown that for a relay node, the packet relaying function is taken care of by the ZNP and does not involve the application processor. For each acknowledgement mode, "PULL" type schemes could be designed, in which a data sink node actively schedules and controls which sensor node should transmit and when. This would require dedicated control signalling messages, which are not desirable for the reasons mentioned before.

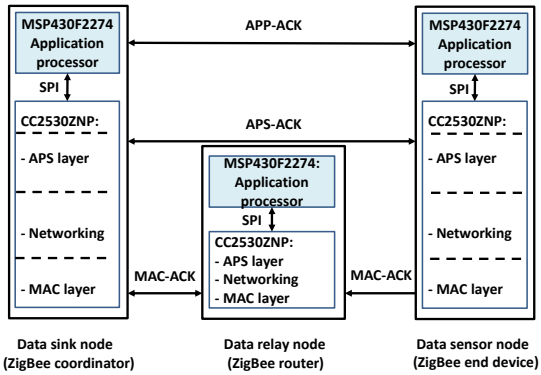


Figure 2. An illustration of three acknowledgement modes.

Therefore, simpler "PUSH" type schemes were adopted as described in the next section.

IV. DATA OFFLOADING SCHEMES

Three schemes are implemented in the system, each based on using one of the three acknowledgement modes.

A. Scheme based on MAC-ACK mode

A sensor node starts to offload its data to the data sink node as soon as it enters the data offload state. After each packet transmission, the ZNP sends the MSP430F2274 processor a positive or negative result depending on whether the MAC layer within the ZNP of the sensor node has received a MAC layer acknowledgment or not.

At the MSP430F2274 of an end device, when a positive result is received, the sensor node waits for a period of T_{app_pos} before sending the next packet. T_{app_pos} must account for, as a minimum, the time required for the data sink node to process a received packet, e.g., sending it through the serial port. When a negative result is received, the sensor node will wait for a period of T_{app_neg} before retransmitting the packet again. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore go into the Sleep state. The data sink node does not perform any scheduling and simply passes whatever it receives to the connected laptop.

This scheme is simple and ideal for the usage scenario where the WSN consists of a data sink node and a single sensor node because acknowledgement is relatively fast in MAC-ACK mode. However, in the case of multiple sensor nodes, there is no guarantee of end-to-end packet delivery, although the reliability can be improved at the expense of increased data offload latency.

B. Scheme based on APS-ACK mode

This scheme works in the same way as Scheme A except that an acknowledgement is from the APS layer. If an APS layer acknowledgement is not received after a period of T_{aps_wait} after a packet being transmitted, the APS layer within the ZNP of the sensor node will retransmit the packet again. After N_{aps_retry} retries without success, the APS layer will

 ISBN: 978-1-61208-391-9

send a negative result to the MSP430F2274. Otherwise, a positive result will be sent to the MSP430F2274.

At the application layer, when a positive result is received, the sensor node waits for a period of T_{app_pos} before transmitting the next packet. T_{app_pos} must account for, as a minimum, the time required for the data sink node to process a received packet. When a negative acknowledgement is received, the sensor node will wait for a period of T_{app_neg} before retransmitting the same packet again. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore go to the Sleep state.

This scheme can provide reliable end-to-end data delivery for the usage scenario where there is only one sensor node, but one or more relay nodes are required to relay packets from the sensor node to the data sink node. In the case of multiple sensor nodes, the end-to-end packet delivery reliability can be improved at the expense of increased data offloading latency, e.g., by increasing the value of T_{app_pos}, T_{app_neg} and/or N_{app_retry}.

C. Scheme based on APP-ACK mode

When this scheme is enabled, the APS-ACK mode will be disabled. A sensor node starts to transmit collected data as soon as it enters the Data Offload state. After a packet is delivered to the ZNP, the MSP430F2274 waits for an application layer acknowledgement for a maximum period of T_{app_wait}. If an acknowledgement is received within T_{app_wait}, the sensor node transmits the next packet after a delay of T_{app_pos}. If no acknowledgement is received when T_{app_wait} expires, the previous transmission is considered to have failed and the same packet is retransmitted after a delay of T_{app_neg}. After N_{app_retry} retries without success, the sensor node will assume that the link to the data sink node has been lost and therefore enters the Sleep state.

The MSP430F2274 of the data sink node issues an acknowledgement for each packet received from a sensor node in addition to sending each received packet to the connected laptop.

This scheme enables reliable end-to-end packet delivery regardless of the number of sensor nodes in the network and network topologies. Like the previous two data offloading schemes, this scheme also uses inter-packet intervals, T_{app_pos} and T_{app_neg}, to regulate traffic from the sensor nodes towards the data sink node. The key difference is that with this scheme the setting of these parameters can be dynamic as well as static. The dynamic setting is achieved by including the values of these parameters in an acknowledgement packet, making this scheme suitable for a range of usage scenarios. For example, when there is just a single sensor node, the parameter, T_{app_pos} for that node could be set to zero so that the sensor node can transmit the next packet as soon as an acknowledgement to the previous packet is received. When there are a number of sensor nodes, the parameter, T_{app_pos} for each sensor node could be set to a different value based on a user configurable priority list in the data sink node.

In summary, Table 1 lists the parameters applicable to the three data offloading schemes.

TABLE I. PARAMETERS APPLICABLE TO EACH SCHEME

Parameters		**Data offloading schemes**		
		MAC-ACK based	***APS-ACK based***	***APP-ACK based***
Parameters withinin MSP430F2274	T_{app_pos}	√	√	√
	T_{app_neg}	√	√	√
	T_{app_wait}			√
	N_{app_retry}	√	√	√
Parameters within ZNP	T_{aps_wait}		√	
	N_{aps_retry}		√	
	T_{mac_wait}	√	√	√
	N_{mac_retry}	√	√	√

Appropriate settings for these parameters depend on the data offloading scheme whilst the optimal data offloading scheme is in turn highly dependent on the usage scenarios. Considering the complex and dynamic nature of the environment in which our system is used, we have implemented our system such that most system operation parameters including the data offloading scheme, threshold level for triggering data collection, accelerometer sampling rate, etc., can be reconfigured on the fly during the system operation.

V. CONCLUSION

We have developed a low cost and adaptive WSN system for collecting raw accelerometer sensor data from railway sleepers. Our system is characterized by built-in adaptive sensor data offloading schemes and a remote control capability so that configurations can be changed after deployment. These features proved valuable for the initial field test involving collecting raw accelerometer data over a section of operational railway line where carrying out data collection reliably and quickly is crucial because accessing an operational rail track is very costly. In the initial field test, the live track was only accessible for a short time and some basic tests were performed. For example, different accelerometer configurations were used to allow the measurement of acceleration at different resolutions for various track conditions, e.g., train approaching, train passing sensor, train passing on adjacent track. The full benefits of our system will be explored in future field trials.

ACKNOWLEDGMENT

This work has been co-funded by Innovate UK.

 ISBN: 978-1-61208-391-9

REFERENCES

[1] L. Angrisani, D. Grillo, R. Moriello, and F. Filo, "Automatic detection of train arrival through an accelerometer," Instrumentation and Measurement Technology Conference (I2MTC), May 2010, pp. 898–902, ISSN: 1091-5281, E-ISBN: 978-1-4244-2833-5.

[2] E. Berlin and K. Van Laerhoven, "Sensor networks for railway monitoring: detecting trains from their distributed vibration footprints," IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS), May 2013, pp.80–87, Print ISBN: 978-1-4799-0206-4.

[3] E. Berlin and K. Van Laerhoven, "Trainspotting: Combining fast features to enable detection on resource-constrained sensing devices," The Ninth International Conference on Networked Sensing Systems (INSS), June 2012, pp.1–8, E-ISBN: 978-1-4673-1784-6.

[4] C. Wang, Q. Xiao, H. Liang, and X. Chen, "On-line vibration source detection of running trains based on acceleration measurement," IEEE/RSJ International Conference on Intelligent Robots and Systems, Oct. 2006, pp.4411-4416, E-ISBN: 1-4244-0259-X.

[5] K. Chebrolu, B. Raman, N. Mishra, P. K. Valiveti, and R. Kumar, "BriMon: A sensor network system for railway bridge monitoring," The Proceedings of the 6th International Conference on Mobile Systems, applications and services, 2008, pp. 2-14, ISBN: 978-1-60558-139-2.

[6] Y. Li and P. Gould, "Embedding Wireless Sensors in Railway Sleepers – Challenges and Choices," Multiple Access Communications Limited White Paper, available from http://macltd.com/publications. [Retrieved: January, 2015].

[7] http://processors.wiki.ti.com/index.php/CC2530ZDK-ZNP-MINI. [Retrieved: January, 2015].

 ISBN: 978-1-61208-391-9

Computer Vision Techniques for Autonomic Collaboration between Mobile Robots

Catherine Saunders, Roy Sterritt, George Wilkie
School of Computing and Mathematics
University of Ulster
Northern Ireland
e-mails: {saunders-c2@email.ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk}

Abstract— **Autonomic Computing is concerned with improving the self-management and adaptability in software. In this paper, we propose an Autonomic concept that would allow robots to interact with each other using an alternative communication protocol in the event that direct wireless digital communications fail. The aim of the research is to create enhanced levels of autonomous behavior that enable the robots to still communicate via a more primitive mechanism. The alternative communication protocol uses a form of textual semaphore. The visual means of communication involves processing textual images that contain meaningful messages or instructions. These textual images are transmitted via a display screen mounted on the robot. The robot receiving the message does so by taking a photo and performing image processing. The research focus is on how robots could adapt to a hardware communications failure by using the technique outlined here. Our experiments have proven that a robot can read text from a screen and respond with an action, e.g., to rotate. Results also indicate that further work is needed in the area of image processing in order to improve the accuracy of the character recognition. It is hoped that these results will lead to further research that can build upon the ideas presented and seek to establish a two-way conversation as opposed to a monologue.**

Keywords-Autonomic Computing; Mobile Robot; Optical Character Recognition; Computer Vision

I. INTRODUCTION

The idea of a self-managing artificial entity is not new; it is a concept that has long existed in Science Fiction. As software systems become more complex, they require more management. As a solution, IBM devised Autonomic Computing [1]. Autonomic Computing seeks to remove human involvement and management of a system [2] and instead allows that system to monitor itself and adapt. Adaption is a fundamental aspect to an Autonomic system, being able to modify its state to cope with external or internal changes is vital to ensure it stays operational.

To be Autonomic or self-managing, a software system should possess awareness of its internal self and external environment. It should also be able to make changes to its state, and to adapt and repair itself [3]. Like the Autonomic Nervous System, an Autonomic system should be able to manage itself without outside influence.

An Autonomic system architecture consists of different layers, each with different responsibilities and awareness [4]. An immediate Reaction layer that can respond to imminent threats to the system; a Routine layer where most processing is performed; a Reflection layer that analyzes past and current experience in order to predict and improve future performance of either the routine or reactive activities [4]. A simple example of a process that could occur in the Reaction layer is that of a mobile robot using on-board sensors, such as ultrasonic or infrared to prevent a collision with an object.

Space robotics is an area that could greatly benefit from software that is Autonomic. The distance between Earth and other planetary bodies makes instantaneous teleoperation of rovers impossible, at least for the foreseeable future. Increasing the Autonomic capabilities of robots so that they may collaborate without the necessity of human intervention would be very advantageous. It would allow a greater number of robots to be sent and enable them to explore unaided.

This paper is part of an ongoing research project that aims to assess and explore areas where Autonomic Computing could be applied in Space Exploration. We discuss how image processing could be used to aid collaboration between entities in the event of a communications failure, i.e., as part of a Self-Healing strategy. Specifically, we look at how a mobile robot could observe and infer meaning from text displayed on the screen of another robot. The purpose is to assess whether meaningful communication can take place and how viable a method Optical Character Recognition (OCR) is for re-establishing communication between entities, where there has previously been a breakdown in the normal radio (wireless) communication. A benefit to having a robot display text as opposed to a symbol or QR code is that a human can also read the message.

Section II of this paper explains why Autonomic Computing is needed and how useful it could be for space exploration. Section III looks at related work involving robotics and text processing. Section IV introduces the research robot platform that was used for the experiments; the setup is also explained through the use of a diagram. Section V gives an overview of the image processing that was used in the experiments. Section VI discusses the algorithm that was used to enable a robot to detect text that is being displayed by another robot in its environment. The experimental results are discussed in Section VII. Section VIII summarizes the paper and discusses future work.

 ISBN: 978-1-61208-391-9

II. RESEARCH BACKGROUND

As a precursor to an eventual human landing on other planetary bodies, probes in the form of satellites and landers have been sent to explore and report. As early as the 1970s, the Soviet Space Program sent teleoperated rovers to the moon [5].

The Mars rover MSL Curiosity is one of many Martian rovers that are paving the way for an eventual human landing. Their overarching goal is to discover as much as possible in order to minimise the risk that is inherent with human space travel. It is hoped that future incarnations of Mars rovers will possess more autonomous behavior, thus making them less dependent on human operators to provide mission direction and navigational waypoints.

NASA's concept missions highlight their interest in eventually sending multiple spacecraft as opposed to one all-important rover. The current process of having a team micromanage a rover's actions would be untenable for a swarm or perhaps even a smaller cluster. The adoption of more ambitious projects like the NASA concept mission Autonomous Nano-Technology Swarm (ANTS) is reliant on the improvement of self-management techniques and the collaboration between multiple self-managing entities [6] [7]. Within a swarm of Autonomic entities, each would need to follow the Self-CHOP paradigm so that the entire swarm can operate efficiently [6][8]. Self-CHOP stands for Self-Configuring, Self-Healing, Self-Optimizing and Self-Protecting; it represents the key functions that an Autonomic system should possess.

Autonomic Computing is concerned with creating software that can self-manage its internal state without the need for human assistance [4][9]. For a system to be considered Autonomic, it must be Self-Aware, Environment Aware, Self-Adjusting and Self-Monitoring. In order to meet these requirements, it needs to monitor its internal and external state, be able to respond to stimuli and update its state. Autonomic Computing research breaks a software system down in to many components known as Autonomic Elements (AE). Each AE consists of an Autonomic Manager (AM) and a Managed Component (MC) that the manager controls via the MAPE control loop [1]. The MAPE control loop was devised by IBM in 2001, the acronym stands for Monitor, Analyze, Plan and Execute [1][10]. The MAPE control loop enables the AM to monitor and adjust the MC.

The AM also monitors the external environment for messages from other AEs. A communications channel exists between the AEs, this ability to exchange information from one AE to another AE is designed to help to improve the overall performance of a system, reduce failure and increase the ability of a system to self-repair.

Information that could be sent from AE to AE via this channel includes a Heartbeat Monitor (HBM) signal, which by its very presence could indicate whether or not another part of the system is still functioning [11][12]. The absence of an *"I am alive"* signal denotes that there is potentially a problem with the AE that is not transmitting. Within a purely software system, there are avenues available for repair and investigation that do not exist with separate mobile robots. With separation, their only means of collaborating as a whole involves transmitting data or communicating via emergence. If there is a fault with their hardware and they are unable to transmit, there is no way for the other members of a swarm or cluster to receive a status report from a faulty robot.

In this paper, we propose that OCR could be used as a backup communication strategy, with robots being able to glean information by reading the screen of a faulty robot. This proposal assumes that not all hardware on the robot is damaged and that it is still able to display messages on its screen. Other work involving OCR and robots has investigated using a standalone robot to observe text in the environment and mapping its location or following textual signs [13][14][15].

III. RELATED WORK

Great advances have been made in robot mapping, object detection and navigation. However, most robots are not equipped with the ability to read. For a robot to operate successfully in a human centric environment, they should be able to fully understand that environment. Analyzing the world around them should not be limited to mapping walls or navigation, as this does not provide a complete picture of where they are. Other forms of visual clue should be considered, perhaps for guidance or directions. Such forms would include reading QR codes or even reading forms of text. There is a limited amount of research on robots making use of OCR; this section will discuss research that combines robotics and OCR.

Using OCR to navigate an indoor environment by reading signs and without using Simultaneous Localization and Mapping (SLAM) has been investigated in [13]. SLAM is a research area that aims to solve the problem of how a mobile robot builds a map of an unknown environment and keeps track of its location within that environment. A robot was equipped with an RFID reader and each sign within the environment had an RF card associated with it. When the reader detected an RF card, a photo was taken and analyzed for text.

In [16], the benefits of using landmarks that occur in the human environment as opposed to artificial landmarks deliberately placed for the robot is discussed. The work presents a technique that enables a robot to recognise a door sign using a Histogram of Oriented Gradients [17] classifiers and then reading the text using Google Goggles [18]. The goal of the research is to create richer maps by gathering semantic information.

The idea that reading text can provide useful semantic information has also been explored in [19]. A robot was set the task of traversing a corridor and reading text of different sizes. It used a pan/tilt/zoom camera to zoom in on potential text. The ability to zoom in on the text improved the accuracy three-fold. *"To date robots have not been able to access this source of semantic information and often rely on*

a second tier of informational infrastructure such as QR codes of RFID tags" [19].

OCR is prone to errors; Poster et al. [20] used probabilistic inference to compensate for detection errors and determine what a detected word was most likely to be e.g., "nqkio" is actually "nokia".

An edge detection algorithm is used by Liu and Samarabandu [21] to better identify characters within an image and allow a mobile robot to navigate. The edge detection algorithm is used to highlight blobs within a photo taken of a sign, bounding boxes are placed around the blobs and used to create sub images, which can then be passed to an OCR engine.

A robot that can read the license plate of a car is discussed in [22], where a mobile robot and Android phone were combined and used a number of image pre-processing techniques to remove noise from the image before OCR recognition took place.

An interesting use of a robot and OCR is presented in [23], the robot operates as a library assistant and can find and fetch books for a user. The user requests a book via a voice command; the robot then uses OCR to locate the book and picks it up using a robotic gripper arm.

Another useful OCR approach is described in [24], where the work involved aiding a blind person by using a shoulder mounted camera system that reads text within the environment and then dictates it to the user. Like the system in [19], it also uses a camera that can zoom in on potential text to get a higher resolution image.

This section has highlighted some examples of research that combines robotics and OCR. The majority of the research in this area tends to focus on improving SLAM and helping a robot navigate an environment by gathering semantic information. This paper presents a novel use for OCR that has not been covered before. We are interested in how two or more robots could communicate via OCR in the event of a communications failure.

IV. ROBOT PLATFORM

The experimentation was completed using the X80 [25] and X80-H [26] models manufactured by Dr Robot. A Samsung Galaxy Tablet was used to display text and mimic a screen that could belong to a robot. The X80 and X80-H are wheeled differential drive style robots that are equipped with a variety of sensors including Ultrasonic and Infrared. For these tests, the Ultrasonic sensors were used for collision prevention with other objects in the environment.

As shown in Figure 1, a separate computer runs the application that connects and controls the robot. The computer is connected via Ethernet to a router and data is sent to and from the robot via Wi-Fi. The application was developed in C# and utilises the Dr Robot API and EMGU CV library [27]. EMGU CV is a .NET wrapper for the widely used Open CV library[28].

The camera included on the X80 is subject to distortion, in future experiments we may opt to use a higher resolution IP WLAN camera that can be placed on the robot. The X80-H robot is equipped with a small mono LCD screen that can display 128x64 pixels Bitmaps.

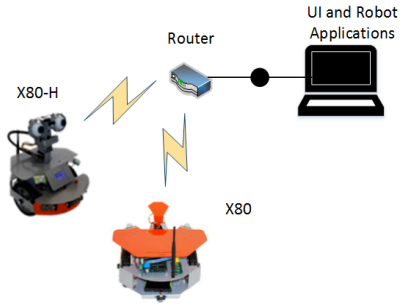


Figure 1. Setup involves robots sending/receiving data via Wi-Fi

We initially planned to use the LCD screen to display text but the X80 was incapable of reading the text due to the dullness of the screen and low quality X80 camera. For this reason, a tablet was chosen instead. Its brighter screen proved easier for the X80 camera to take images, from which text could be detected.

V. IMAGE PROCESSING

Image processing is a vast area of research within mobile robotics; its primary focus is to help with SLAM. However, the idea that robots could communicate by reading each other's screens is not a topic that receives much attention. This is because it is a relatively inefficient way to communicate compared to more traditional methods such as wireless communication, Infrared and Bluetooth. It is, however, interesting to discuss different approaches like OCR, or even speech recognition were messages broadcast from a speaker on a robot could be processed for words by another robot, because such means of communication may provide a useful backup system in the event that the primary mode of communications fails – thereby fitting our goal of creating adaptability within our robot capabilities.

For the experiments, we used the Tesseract library, which was initially developed by HP Labs and is currently maintained by Google [29]. Processing and analyzing the data within each video frame is computationally inefficient. Using color filtering and feature based matching reduces the amount of processing required. By only checking for characters within a Region of Interest (ROI), this eliminates the reading of extraneous data that may occur in the environment. To create a ROI, such as a rectangle, we can place a red rectangular frame around text; everything except the color red can then be filtered out. The next step is to look for contours with 4 edge points, draw a bounding box around the contour, and clip the X, Y, Width and Height of this bounding box from the original frame. The Tesseract OCR Engine is then used to look for text within this clipping and not the entire frame. By only processing the characters that appear within the red bounding box, this helps to cut down on the amount of false data that could be read.

 ISBN: 978-1-61208-391-9

VI. PROTOYPE SYSTEM

We devised an algorithm that enables a robot to check on another robot if the *"I am alive"* signal has not been received. The experiment is conducted under the premise that if the X80 robot has not received the signal, it then moves towards the X80-H in order to investigate.

The flow diagram in Figure 2 shows the steps taken if the signal is not received. The robot that has noticed the signal is absent moves to the last known co-ordinates of the robot that is not transmitting. When encountering an object <0.3m it processes a frame of its video feed and looks for red contours with 4 points, if it finds a contour, it looks for text within the bounding box of those 4 points. The purpose of this is to test whether two robots could still communicate information if they no longer have access to the more usual means of wireless communication due to a hardware fault or atmospheric conditions. We used red contours but other shapes and colors could be used to help recognise the screen of the robot that is not transmitting. Another idea would be to train a Haar classifier and have the robot look for the exact image of the non-transmitting robot in its video feed.

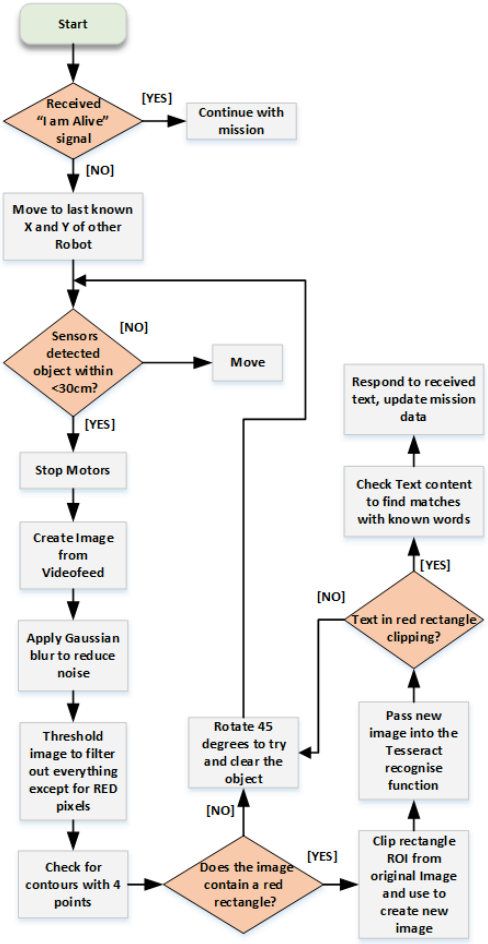


Figure 2. Flow diagram of OCR communication

The Algorithm in Figure 3 shows the steps taken when a robot does not receive the *"I am Alive"* heartbeat signal from another robot within a cluster. The robot moves to the last known co-ordinates and takes a photo, checks the image or text and then follows the command associated with the text.

Algorithm: If Heartbeat signal is not received

```
1:  Robot autonomously moves around the environment
2:  if "I am Alive" signal has not been received then
3:     Move to last known co-ordinates of other robot
4:        if sensor detects object within 30cm then
5:           Stop movement
6:           Take snapshot of video frame
7:           Look for red rectangles within the snapshot
8:        end if
9:        if red rectangles exist in snapshot then
10:          if rectangle contains text then
11:             if text matches a command word then
12:                Follow Command e.g., Danger
13:             end if
14:          end if
15:       end if
16: end if
```

Figure 3. Algorithm used when a heartbeat signal is not received

VII. EXPERIMENTAL RESULTS

The tests involved an X80 and an X80-H robot, with the latter being assigned the role of sender. A Samsung Galaxy Tablet was partially covered in red card and placed on the X80-H, this was used to display the word "DANGER". The X80 was then tasked with approaching the X80-H, stopping when it detected an object <0.3m and using image processing techniques to find a red rectangle and looking for text within it.

In Figure 4, we can see the X80 (right hand robot) has stopped approximately 0.3m from the X80-H (left hand robot in the picture). The X80-H robot has the tablet positioned at the front of its base. The tablet has text displayed; the red rectangle ensures that only the text within it will be processed by the X80

Figure 4. Experiment using an X80-H, X80 and Tablet

In Figure 5, the 1st row from left to right displays the Robot Camera video feed, the 2nd image is of a screenshot taken of the video feed, the blue bounding box highlights a

detected red rectangle. The Red Filter image box is a gray scale representation of the screenshot, a threshold filter has been applied so that only red pixels will display in white, non-red pixels are shown in black. The 2nd row shows the OCR output with the word "DANGER" detected; the Rectangle's image box gives a clearer indication of the size of the rectangle detected. The final image is of the Clipped Rectangle bounding box, it is this image that is used in conjunction with the Tesseract library. Only this image is checked for text.

We can see that only the inner rectangle has been detected, the outer rectangle was occasionally detected; however this made no difference to whether the text was correctly identified. The Red Filter image box shows that the inner rectangle has a more even shape due to the white inner box. It has been noticed that the camera tends to classify non-background pixels as red; this then results in a somewhat fuzzy appearance of the outer box. Due to the irregular shape the outer box has more than 4 contour points and is therefore not detected as a rectangular object.

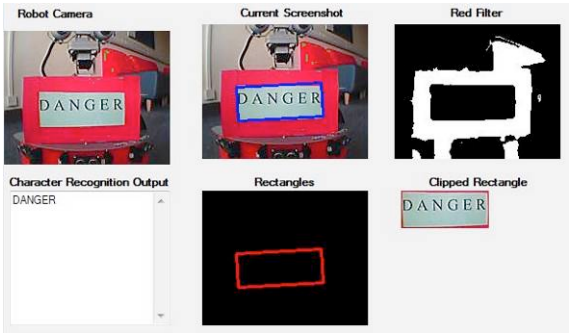


Figure 5. GUI showing processed image and detected rectangle

A. Experiment 1 Results

Table I shows the results of 10 tests where the receiver robot was approximately 0.29 metres from the sender robot. The sender robot or X80-H is shown on the left in figure 3. The test was performed under optimum lighting conditions with artificial and natural light. The red rectangle was detected 100% of the time; the OCR output included the word "DANGER" 40% of the time. In 2 cases, the OCR failed to detect any characters, the other 40% included variations on the word, with "7ANGER" being detected 30% of the time.

The X and Y co-ordinates of the inner rectangle are displayed in the table, as are the Width and Height, the readings were very consistent. The distance between the two robots was the same throughout each test, the light in the room varied slightly from one test to the next. The reason for the failed detections as the tests go on seems to lie with the slight variation in lighting. More work is required in order to improve character detection under poor lighting conditions.

TABLE I. RESULTS FROM EXPERIMENT 1

	Detect Rectangle	OCR Output	X	Y	W	H
1	Yes	DANGER	36	61	92	42
2	Yes	---	36	61	92	42
3	Yes	DANGER	36	61	92	42
4	Yes	DANGER	36	62	92	41
5	Yes	DANGER	36	62	92	41
6	Yes	7ANGER	37	60	91	43
7	Yes	7ANGER	37	60	91	43
8	Yes	7ANGER	37	60	91	43
9	Yes	---	37	61	91	41
10	Yes	7ANOER	37	60	91	43

B. Experiment 2 Results

For this experiment, the robot was programmed to read the text within a red rectangle and then respond with an action. If the OCR output contained the word "DANGER" or variations of it, the robot would rotate 360°, results of the experiment are displayed in Table II. The variations observed in Experiment 1 were incorporated in to the code for Text checking in Experiment 2. This test was undertaken to show that the robot could interpret detected text and respond with an action.

TABLE II. RESULTS FROM EXPERIMENT 2

	Detect Rectangle	OCR Output	X80 Action
1	Yes	%^NGER	NONE
2	Yes	DANGER	ROTATED
3	Yes	DAWN	NONE
4	Yes	---	NONE
5	Yes	DANGER	ROTATED
6	No	---	NONE
7	Yes	DANGER	ROTATED
8	Yes	DANGER	ROTATED
9	Yes	DANGER	ROTATED
10	Yes	DANGER	ROTATED

Checking whether a misspelt detected word is similar to a correctly spelled word was explored in [20], where the researchers were able to improve text detection accuracy by using probabilistic inference. Our experiments have shown that more work is required in image pre-processing to correct distortion before an image is passed to the OCR engine. The experiments were conducted at a distance of approximately 0.29 metres; this was due to limitations with the camera being used. In [19][24], the researchers used a pan/tilt/zoom camera to zoom in on a region of interest. In

 ISBN: 978-1-61208-391-9

future we would like to try a similar approach as this would allow for experiments to be conducted at a greater distance. Future experiments will also seek to include more meaningful instructions and responses.

C. *Further Experiments*

Further experiments were carried out that tested the X80 reading the screen at various angles, the rectangle was repeatedly detected but unfortunately the word could not be recognised unless the robot was facing the screen straight on. The variation observed when lighting conditions are sub optimal can be seen in Figure 6; in this case, too much sunlight has caused red pixels to be detected as green pixels. The Red Filter is unable to detect a rectangle and therefore cannot look for text. However, this is also due to the X80's camera; it has proven to be of low quality and regularly experiences distortion. In future, similar experiments could be performed with a higher resolution camera equipped with a zoom capability.

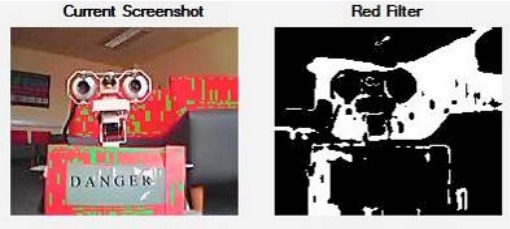


Figure 6. X80 camera distortion, red pixels appear green

Numerous other tests were conducted that involved having a robot read printed words. Using printed text as opposed to an electronic screen greatly improved the rate of detection; this is because the Tesseract OCR has been developed for use with printed text. These tests involved the X80 robot moving around a room and checking for text when it encountered an object < 0.3m away. The text contained various words that could be translated in to simple commands such as Rotate, Move Right, Reverse and Stop. Figure 7 shows the results from a test where the robot was instructed only to process text contained within a red rectangle.

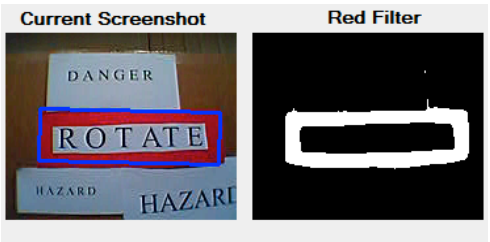


Figure 7. Finding a red rectangle

Figure 8. Camera distortion due to lighting conditions

Despite the higher success rate observed with the printed word test, camera distortion still caused a high failure rate. Figure 8 shows an example of the level of distortion the camera is prone to experiencing. The noise tends to distort the shape of the characters and prevent a word being detected accurately.

VIII. SUMMARY AND FUTURE WORK

In this paper, we proposed a concept that would allow mobile robots to communicate using a form of textual semaphore. By locating an object using Ultrasonic sensors, a robot can analyze whether there is a red rectangle within its field of view. The red rectangle frames a screen used to display textual commands. The contents of a detected rectangle are processed by the responding robot. If the rectangle contains text, then the robot has found the non-transmitting robot. The experiments showed that lighting conditions and the low quality camera greatly affected the results; future experimentation would look at how to compensate for this so that the text can still be read. Improving image noise by using pre-processing techniques will also be explored in future.

Future experimentation may involve instructing one robot to follow another and read the directions on the screen. The robots could be placed within a text-rich environment to test how well the method works. Further work is also needed to see if the text detection rate can be improved and poor lighting conditions compensated for. We would also like to test different screens to ascertain whether some are easier to read at an angle than others. Performing future experiments with a high resolution IP camera instead of the video camera supplied with X80 would help eliminate unwanted image distortion.

The end goal of this research is to develop a system that allows robots to converse via screens with changeable text. Our results to date are encouraging and demonstrate that the principle of communicating in this manner has some potential.

REFERENCES

[1] IBM, "An Architectural Blueprint for Autonomic Computing," IBM, 2003.

[2] R. Sterritt, "Towards Autonomic Computing : Effective Event Management," in Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW-27'02), 2003, pp. 40–47.

[3] R. Sterritt and D. Bustard, "Towards an autonomic computing environment," in Proceedings of the 14th International Workshop on

Database and Expert Systems Applications (DEXA'03), 2003, pp. 694–698.

[4] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A Concise Introduction to Autonomic Computing," in Advanced Engineering Informatics, 2005, vol. 19, no. 3, pp. 181–187.

[5] L. Matthies et al., "Computer Vision on Mars," in International Journal of Computer Vision, 2007, vol. 75, no. 1, pp. 67–92.

[6] R. Sterritt, C. Rouff, J. Rash, W. Truszkowski, and M. Hinchey, "Self-* Properties in NASA Missions," in 4th International Workshop on System/Software Architectures (IWSSA'05) in Proceedings of the International Conference on Software Engineering Research and Practice (SERP'05), 2005, pp. 66–72.

[7] R. Sterritt and M. Hinchey, "Engineering Ultimate Self-Protection in Autonomic Agents for Space Exploration Missions," in 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, 2005, pp. 506–511.

[8] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, "NASA's Swarm Missions: The challenge of Building Autonomous Software," IEEE IT Pro, 2004, vol. 6, no. 5, pp. 47–52.

[9] A. G. Ganek and T. A. Corbi, "The Dawning of the Autonomic Computing Era," IBM Systems Journal, 2003, vol. 42, no. 1, pp. 5–18.

[10] B. Jacob, R. Lanyon-hogg, D. K. Nadgir, and A. F. Yassin, A Practical Guide to the IBM Autonomic Computing Toolkit. IBM, 2004.

[11] M. G. Hinchey and R. Sterritt, "Self-Managing Software," IEEE Computer, 2006, vol. 39, no. 2, pp. 107–109.

[12] R. Sterritt and D. F. Bantz, "Personal autonomic computing reflex reactions and self-healing," in IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 2006, vol. 36, no. 3, pp. 304–314.

[13] S. N. Shaikh and N. V Londhe, "OCR Based Mapless Navigation Method of Robot," International Journal of Inventive Engineering and Sciences, 2013, vol. 1, no. 8, pp. 6–12.

[14] C. Case, B. Suresh, A. Coates, and A. Y. Ng, "Autonomous sign reading for semantic mapping," IEEE International Conference on Robotics and Automation, 2011, pp. 3297–3303.

[15] D. Létourneau, F. Michaud, J. Valin, and C. Proulx, "Making a Mobile Robot Read Textual Messages," In Proceedings IEEE Conference on Systems, Man, and Cybernetics, 2003, vol. 5, pp. 4236-4241.

[16] J. G. Rogers, A. J. B. Trevor, C. Nieto-Granda, and H. I. Christensen, "Simultaneous localization and mapping with learned object recognition and semantic data association," in IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 1264–1270.

[17] N. Dalal and W. Triggs, "Histograms of Oriented Gradients for Human Detection," in Computer Vision and Pattern Recognition, CVPR., 2005, vol. 1, pp. 886–893.

[18] "Google Goggles." [Online]. Available: http://en.wikipedia.org/wiki/Google_Goggles. [Accessed: 24-Jan-2015].

[19] M. L. Wyss and P. Corke, "Active Text Perception for Mobile Robots," in unpublished, 2012.

[20] I. Posner, P. Corke, and P. Newman, "Using text-spotting to query the world," in IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems, IROS 2010 - Conference Proceedings, 2010, pp. 3181–3186.

[21] X. Liu and J. Samarabandu, "An Edge-based Text Region Extraction Algorithm for Indoor Mobile Robot Navigation," in IEEE International Conference Mechatronics and Automation, 2005, pp. 701–706.

[22] H. F. Chen, C. Y. Chiang, S. J. Yang, and C. C. Ho, "Android-based patrol robot featuring automatic license plate recognition," in Computing, Communications and Applications Conference, 2012, pp. 117–122.

[23] R. Ramos-Garijo, M. Prats, P. J. Sanz, and A. del Pobil, "An autonomous assistant robot for book manipulation in a library," in SMC'03 Conference Proceedings. IEEE International Conference on Systems, Man and Cybernetics. Conference Theme - System Security and Assurance, 2003, vol. 4, pp. 3912–3917.

[24] N. Ezaki, M. Bulacu, and L. Schomaker, "Text Detection from Natural Scene Images: Towards a System for Visually Impaired Persons," in 17th International Conference on Pattern Recognition (ICPR), 2004, vol. 2, pp. 683–686.

[25] "Dr Robot X80." [Online]. Available:http://www.drrobot.com/products_item.asp?itemNumber=X80. [Accessed: 24-January-2015].

[26] "Dr Robot X80-H." [Online]. Available:http://www.drrobot.com/products_item.asp?itemNumber=X80-H. [Accessed: 24-January-2015].

[27] "EMGU CV." [Online]. Available:http://www.emgu.com/wiki/index.php/Main_Page. [Accessed: 24-January-2015].

[28] "OpenCV (Open Source Computer Vision)." [Online]. Available: http://opencv.org/. [Accessed: 24-January-2015].

[29] Google, "Tesseract-OCR." [Online]. Available:https://code.google.com/p/tesseract-ocr/. [Accessed: 24-January-2015].

 ISBN: 978-1-61208-391-9

Sink Mobility Strategies for Reliable Data Collection in Wireless Sensor Networks

Yuki Fujita*, Daichi Kominami† and Masayuki Murata*
*Graduate School of Information Science and Technology, Osaka University, Japan
{y-fujita, murata}@ist.osaka-u.ac.jp
†Graduate School of Economics, Osaka University, Japan
d-kominami@econ.osaka-u.ac.jp

***Abstract*—The Internet of Things and machine-to-machine communications will form one of the most important backbones of our life in the near future. Therefore, more reliability is required in many wireless sensor network applications, such as structural health monitoring systems, intruder detection systems, and search and rescue systems. However, without the assumption that all sensor nodes can reach a sink node through multi-hop communication and that the connectivity among all sensor nodes is stable, it is difficult to guarantee the reliability of data collection. In this paper, we focus on controlling the mobility of a mobile sink and propose two types of mobility strategies to collect sensing data certainly from all sensor nodes in an observed area. One strategy is to learn the positions of all isolated networks in the observed area and the other is to collect sensing data using the learned positions. Through computer simulations, we show that the mobile sink with the mobility strategies can collect the sensing data from all sensor nodes.**

Keywords–Wireless sensor networks, reliable data collection, mobile sink, controlled mobility.

I. INTRODUCTION

Supporting assured data collection in wireless sensor networks (WSNs) is one of the significant challenges in frequently changing environments. This is because dynamic changes in the observed area and loss of reachability to sink nodes occur in actual situations, which cannot be dealt with through conventional transport techniques. This promotes network-level reliable mechanisms for data collection. We focus on the mobility control for a data collecting node, usually called a *sink node* in WSNs, to realize reliable data collection.

Wireless sensor networks, which facilitate the collection of environmental information, are expected to apply significantly to various applications, e.g., structural health monitoring of infrastructures, monitoring of temperature and humidity on a farm, tracking of animals, etc. [1], [2]. In many cases, WSNs are composed of many sensor nodes and a few sink nodes, which operate in a distributed manner and are connected to each other. Sensor nodes forward their sensing data to one of the sink nodes through multi-hop wireless communications, which makes it possible to collect various environmental information.

In the near future, many machines will be mutually connected and will be quietly embedded in our life space. Thus, the Internet of Things and machine-to-machine communications will make WSN techniques more and more significant. In applications strongly tied to safety and security, the reliability of data gathering is one of the most important viewpoints. Data collection is realized under the assumption that all sensor nodes are reachable by one of the sink nodes through multi-hop communication. However, this assumption is not always realistic due to the limitation of the communication range of nodes, changes of wireless channel conditions, or failures of sensor nodes. It is inappropriate to allow nodes to directly communicate with a sink node since sensor nodes have a limited battery capacity in most WSNs. Also, it is difficult to deploy sensor nodes over the observed area with paying excess attention that all sensor nodes are always reachable to a sink node.

We focus on a sink node with mobility called a *mobile sink*. A mobile sink can achieve both reduction of power consumption and reachability of every sensor nodes by approaching each sensor node, receiving data, and carrying it to the static base station. Many studies have been conducted about mobile sinks as a solution for power saving, which is one of the challenging problems in WSNs, such as path planning of mobile sinks and efficient data routing algorithms considering the movement of a mobile sink [3]–[5]. Mobile sinks from the viewpoint of reliable data gathering do not have as much active research.

Controlled mobility is a key idea for maintaining network connectivity and achieving data reachability for users, where the mobility of mobile sinks is dynamically controlled from both inside and outside of networks [6], [7]. We previously combined controlled mobility with a potential-based routing mechanism in a wireless sensor network, where periodically transmitted route information messages lead a mobile sink toward the static data collecting node (called the target node) and the mobile sink receives all data from the target node [8]. In this method, a mobile sink can collect data from a network, however, the mobile sink cannot deal with the loss of reachability to the target node.

In this paper, we propose two mobility strategies for a mobile sink for realizing reliable data collection. To this end, we need to manage the following two changes in an observed area caused by failures, energy depletion, or additions of sensor nodes. Here, we define a *sub-network* as a set of all sensor nodes reachable from each other by multi-hop communication.

- Small changes in a sub-network, which do not increase

 ISBN: 978-1-61208-391-9

or decrease the number of sub-networks, but cause changes in route. This occurs mainly due to link failures and node failures.

- Large changes in a sub-network, which increase or decrease the number of networks and also cause changes in route. This occurs for various reasons that cause the disconnection of a sub-network, the jointing of sub-networks, or the deployment of a new sub-network.

Small changes have been well-studied, however, large changes have been little considered in existing studies. Thus, our interests are in how we can collect *all* data in an observed area when both types of changes occur. We use the term *reliable data collection* as 100% collection of data that are generated by all sensor nodes in each sub-network. We aim for reliable data collection by using a mobile sink within an observed area where both the number and the positions of sensor nodes are unknown. In these situations, it is a possible (but not practical) method for a mobile sink to travel all over the observed area since the mobile sink does not know where sub-networks are in the observed area. Of course, it takes much more time for the mobile sink to travel to every nook and cranny as the observed area gets larger. Therefore, the first mobility strategy is conducted at long intervals, where a mobile sink travels all over the observed area to grasp all positions of sub-networks and also impassable locations. The other strategy is to visit all sub-networks and to collect sensing data from data possessing nodes using learned positions from the first strategy.

In principle, it is difficult to catch an unexpected change by methods other than the first strategy, and it requires a lot of time. Therefore, it is taken repeatedly over a long period. For the second strategy, we use a clustering technique and all data in a sub-network are gathered in one or more cluster heads. Then, a mobile sink just has to visit such cluster heads to collect data. Note that we assume that cluster heads change their role back to a non-cluster head periodically for managing small changes in a sub-network and for achieving load balancing. Thus, a mobile sink does not always know the position of a cluster head. The controlled mobility mechanism proposed in [8] helps a mobile sink approach a cluster head in each sub-network.

The remainder of this paper is organized as follows. In Section II, we present the mobility control strategy for memorizing locations of networks. In Section III, we show the mobility strategy for collecting sensing data in a network. Section IV presents simulation results, and finally, we conclude our paper in Section V.

II. Mobility Strategy for Memorizing Sub-Network Locations

A mobile sink has to periodically check the entire picture of the observed area, such as the positions of sub-networks and forbidding places, to determine the path for visiting all sub-networks. In order to grasp this information, it moves over the entire observed area while identifying and memorizing all the different sub-networks.

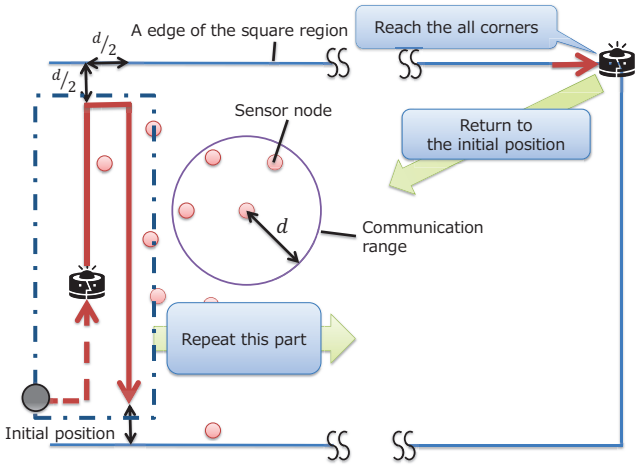


Figure 1. Mobility strategy for detecting all sensor nodes without any oversight

In our proposal, a mobile sink moves so that it does not overlook even one sensor node placed in the observed area. To begin with, we assume that only a mobile sink has a positioning device like a global positioning system. A mobile sink commences to move from a given initial position (e.g., a base station with charging capability for a mobile sink battery), which is one of the corners of the pre-defined square region including the whole observed area as illustrated in Fig. 1. Then, the mobile sink goes straight on toward one nearby corner until it reaches $d/2$ length short of the corner, where d is the wireless communication range of the mobile sink and sensor nodes. Then, it takes a turn toward the other nearby corner, moves ahead $d/2$, and again rotates in the same angle and moves ahead. The mobile sink repeats the same process until reaching all corners, then it returns to the initial position.

In this strategy, a mobile sink intercepts a message **PInfoMsg**, which all sensor nodes transmit and exchange with each other for updating route information (described in Section III-B in detail). Then, the mobile sink acquires a special identifier (ID) contained in that **PInfoMsg**, which is used to identify sub-networks.

A mobile sink memorizes or updates a position of sub-network according to Algorithm 1, where some terms are listed in Table I. A mobile sink has a table **NetTable** for storing sub-network positions, which is updated every time it receives a **PInfoMsg**. A **NetTable**'s entry is the tuple ($netID$, $position$, RSSI) where $netID$ is an identifier of a sub-network, $position$ is the position where the mobile sink received the **PInfoMsg**, and RSSI is the received signal strength indication of the **PInfoMsg**. A new entry is always registered to the table if there is no entry whose $netID$ equals one in the **PInfoMsg**, and an existing entry is updated if the RSSI of the received **PInfoMsg** is greater than an existing one that has the same $netID$ of the **PInfoMsg**. This is for ensuring that the mobile

 ISBN: 978-1-61208-391-9

Algorithm 1 Memorizing the positions of networks associating with $netID$ by the mobile sink

```
// The mobile sink moves in every corner of the observed area.
repeat
  if intercepts PInforMsg(i, netID_i, pList_i, vList_i) then
    if NetTable has no entry with netID_i then
      register NetTable(netID_i, pos, PInforMsg.rssi)
    else
      if PInfoMsg.rssi > entry.rssi then
        update NetTable(netID_i, pos, PInforMsg.rssi)
      end if
    end if
    if State_i is CLUSTER(i, 0) then
      sends SensingDataRequest to S_i
    end if
  end if
until reaches the end of the observed area
```

TABLE I. NOTATIONS IN OUR PROPOSED METHODS

Notation	Description
N_s	The number of sensor nodes which is initially deployed
N_b	The number of sensor nodes which will get failed
N_a	The number of sensor nodes which will be added
S_i	The sensor node whose ID is i
$ND(S_i)$	The number of neighbor nodes of S_i
$State_i$	The state of S_i which indicates whether S_i belongs to a cluster or not. $State_i$ is either **UNCLUSTER** or **CLUSTER**(i, n)
t_{S_i}	The current time of node S_i
T_{lim}	The time limit to search for its neighbor nodes
T_{flood}	The interval that cluster heads broadcast **PInfoMsg**
T_h	The period that **NTable** keeps a non-updated $entry$
$netID_i$	The ID of the network that S_i belongs to
pID	The ID of the potential field
myP_i	The potential values that S_i has
$pList_i$	The set of pID that S_i has
$vList_i$	The set of myP_i that S_i has

sink can obtain more accurate positions of sub-networks. Note that, when the mobile sink can contact a cluster head in this mobility strategy, it demands sensing data from the cluster head by transmitting a message **SensingDataRequest**.

III. MOBILITY STRATEGY FOR VISITING ALL CLUSTER HEADS

In the previous strategy, a mobile sink memorizes the position of all sub-networks. In order to realize reliable data collection, the mobile sink has to visit each sub-network, collect all sensing data in the sub-network, and then move ahead to another sub-network. One way for collecting all sensing data in a sub-network is to visit sensor nodes one by one by memorizing the position of each node in the previous strategy, which spends memory costs and time costs. In our proposal, all sub-networks within the observed area have one or more special nodes that gather sensing data from all sensor nodes in an individual sub-network, and a mobile sink moves to and sojourns with them to bring the sensing data to the initial position. We elect this special node by using a cluster head election algorithm proposed in [9]. When a mobile sink enters a sub-network, the mobile sink intercepts and interprets route information messages exchanged with sensor nodes and moves in the same direction as data flows, and consequently, it can visit all cluster heads in the sub-network. After that, the mobile sink moves ahead in the direction of another sub-network learned at the previous strategy.

A. Cluster heads election

We elect one or more cluster heads for each sub-network by using a part of the DEECIC algorithm [9] with minor modification. The cluster head election algorithm in DEECIC is described in Algorithm 2 with some terms which are tabulated in Table I. First, sensor node S_i broadcasts an **UpdatePacket** to notify its neighbors of its presence at randomly chosen time t $(0 < t < T_{lim})$. Then, S_i broadcasts a **DegreePacket** including $ND(S_i)$, which is the number of received **UpdatePackets** until T_{lim} expires, to inform its node of its degree at t $(T_{lim} \leq t < T_{lim} + T_{S_i})$. T_{S_i} is given from $T_{S_i} = \alpha e^{1/ND(S_i)}$ where α is a constant so that it ensures $0 < T_{S_i} \ll T_{lim}$. Sensor node S_i waits for receipt of a **StatePacket** including the state of a neighbor node, which notifies whether the neighbor belongs to any cluster or not. Here, **CLUSTER**(i, n) presents that the node S_i can reach a certain cluster head within n hops and CLUSTER$(i, 0)$ means that node S_i is a cluster head. Upon receiving a **StatePacket** with **CLUSTER**(s, n), S_i sets $State_i$ to **CLUSTER**$(i, n + 1)$ if $State_i$ was **UNCLUSTERD** or **CLUSTER**(i, m) where m is larger than $n + 1$. S_i broadcasts a **StatePacket** if $n + 1 \leq max\_n$ thereafter, which limits the coverage of each cluster.

When disconnection of a sub-network occurs, there may be no cluster head in a sub-network. Therefore, this cluster election is done periodically. Thus, our proposed method can respond to environmental changes inside a sub-network.

B. Construction and update of potential fields

We use a potential-based routing protocol [10] for data collection in each sub-network. The potential-based routing is known as a resilient routing protocol for environmental variations. Every node updates its own potential, which is a scalar value calculated only with local information—own potential, neighbors' potential, and node degrees. It is worth noting that potential messages exchanged between sensor nodes for data routing is also utilized for the guidance of a mobile sink toward elected cluster heads.

For the mobility strategy to visit all cluster heads, every cluster head constructs one potential field that is the shape of a concave curve whose bottom corresponds to the position of the cluster head. Thus, multiple potential fields can be constructed in each sub-network. Each potential field has a unique identifier pID that corresponds to an identifier of the

 ISBN: 978-1-61208-391-9

Algorithm 2 Selection of cluster heads in a network

1: // all nodes perform following;
2: $State_i \Leftarrow$ **UNCLUSTER**
3: $t \Leftarrow$ random value between 0 and T_{lim}
4: broadcast **UpdatePacket** at t_{S_i}
5: **repeat**
6: **if** receives a **UpdatePacket then**
7: $ND(S_i) \Leftarrow ND(S_i) + 1$
8: **end if**
9: **until** $t_{S_i} \geq T_{lim}$
10: broadcast **DegreePacket** at t, $(T_{lim} \leq t < T_{lim} + T_{S_i})$
11: **repeat**
12: **if** receive **StatePacket** with **CLUSTER**(s, n) node **then**
13: **if** $State_i$ is **UNCLUSTER then**
14: $State_i \Leftarrow$ **CLUSTER**$(s, n+1)$
15: **else if** $State_i$ is **CLUSTER**(i, m) and $m > n+1$ **then**
16: $State_i \Leftarrow$ **CLUSTER**$(s, n+1)$
17: **end if**
18: **if** n+1 $\leq max\_n$ **then**
19: broadcast **StatePacket**
20: **end if**
21: **end if**
22: **until** $t_{S_i} \geq T_{lim} + T_{S_i}$
23: **if** $ND(S_i)$ is larger than all neighbor nodes and $State_i$is **UNCLUSTER then**
24: $State_i \Leftarrow$ **CLUSTER**$(i, 0)$
25: broadcast **StatePacket**
26: **end if**

cluster head that is a owner of the potential field. The potential-field construction process is given in Algorithm 3 with some terms tabulated in Table I.

First, cluster head S_i initializes $netID_i$ and myP_i, and then broadcasts a **PInfoMsg** throughout the sub-network (lines 1–10). A **PInfoMsg** includes the sender's ID (S_i), sub-network ID, multiple potential-field IDs (pID), and potential values of correspondent potential fields as illustrated in Fig. 2. When receiving a **PInfoMsg**, a sensor node updates the **NTable**, myP_i, and $netID_i$ and broadcasts a new **PInfoMsg** (lines 11–44). Here, **NTable** is a table to store information about the potential of neighbor nodes, and its entry is composed of five attributes $(src, networkID, pID, pValue, time)$, which are an ID of the source node (cluster head), a sub-network ID, a potential-field ID, a potential value in the correspondent potential field, and the time this entry was registered or updated, respectively. S_i registers or updates an entry of their **NTable** when receiving a **PInfoMsg** and S_i removes an old entry that is not updated for a period T_h. After that, S_i calculates its own sub-network ID ($netID_i$) and potentials (myP_i). $netID_i$ is set to the lowest value among pID registered in its **NTable** and myP_i is set to an average potential value of its neighbors as proposed in [10]. Finally, S_i updates all myP_i and puts them into a **PInfoMsg**. After a lapse of $T_{forward}$, it broadcasts its own **PInfoMsg**.

Algorithm 3 Potential fields construction and update

1: **if** $State_i$ is **CLUSTER**(i,0) **then**
2: $netID_i \Leftarrow i$
3: $myP_i[i] \Leftarrow initial\_potential$
4: puts i into $pList_i$
5: puts $myP_i[i]$ into $vList_i$
6: broadcast **PInfoMsg**$(i, netID_i, pList_i, vList_i)$ per T_{flood}
7: clear $pList_i, vList_i$
8: **else**
9: $netID_i \Leftarrow NULL$
10: **end if**
11: **loop**
12: **if** receive **PInfoMsg**$(s, netID_s, pLIst_s, vLIst_s)$ **then**
13: **for** $j = 1$ to k **do**
14: update the **NTable**$(s, netID_s, pList_s[j], vList_s[j], t_{S_i})$
15: **end for**
16: **for all** $entry$ in **NTable do**
17: **if** $t_{S_i} - entry.time > T_h$ **then**
18: remove the $entry$ from **NTable**
19: **if NTable** has no $entry$ **then**
20: $State_i \Leftarrow$ **CLUSTER**$(i, 0)$
21: broadcast **StatePacket**
22: **else if NTable** has no $entry$ whose $entry.netID$ is $netID_i$ **then**
23: **if** $State_i$ is **CLUSTER**$(i, 0)$ **then**
24: $netID_i \Leftarrow i$
25: **else**
26: $netID_i \Leftarrow NULL$
27: **end if**
28: **end if**
29: **end if**
30: **end for**
31: **for** $j = 1$ to k **do**
32: update all $myP_i[pList_s[j]]$
33: **end for**
34: **if** $netID_i$ is $NULL$ or $netID_i > netID_s$ **then**
35: $netID_i \Leftarrow netID_s$
36: **end if**
37: **for** $j = 1$ to k **do**
38: puts $pList_s[j]$ into $pList_i$
39: puts $myP_i[pList_s[j]]$ into $vList_i$
40: **end for**
41: broadcast **PInfoMsg**$(i, netID_i, pList_i, vList_i)$ after $T_{forward}$
42: clear $pList_i, vList_i$
43: **end if**
44: **end loop**

C. Traveling to cluster heads according to potential fields

Each cluster head has a unique potential field and also has the minimum potential value in its potential field. All sensor nodes forward their sensing data to one of their neighbor nodes

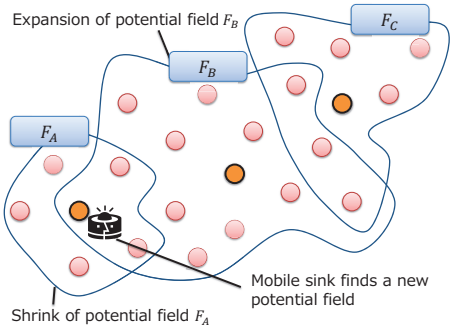


Figure 4. New potential field F_B is detected by the mobile sink after potential field F_A decreases its potential

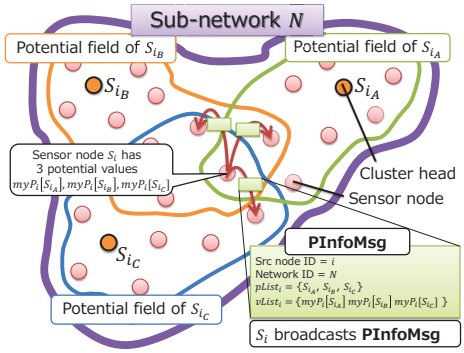


Figure 2. Example of a situation where S_i broadcasts a **PInfoMsg**

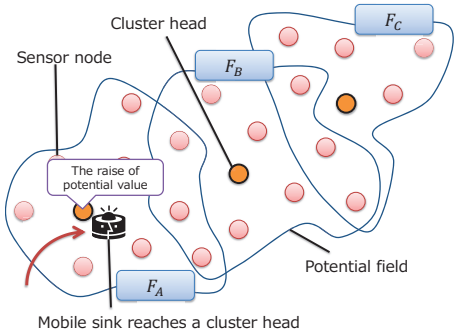


Figure 3. Increase of a potential value of a cluster head caused by arrival of the mobile sink

that have a smaller potential value, and then, all sensing data eventually reach one of the cluster heads. The mobile sink also utilizes these multiple potential fields, that is, utilizes $vList$ in a **PInfoMsg** transmitted by a sensor node to reach a cluster head [8]. After the arrival of the mobile sink at a cluster head, the cluster head gives collected data to the mobile sink and increases its potential value greatly. This decreases the priority of the potential field of the cluster head compared with other potential fields because a sensor node with a "smaller" potential value attracts data and the mobile sink as shown in Fig. 3. After that, the mobile sink can find a new potential field and can go to another cluster head as shown in Fig. 4.

When a mobile sink has visited all cluster heads in a sub-network, the mobile sink goes to an unvisited sub-network. Then, the mobile sink visits all sub-networks in the order that it previously visited and memorized them in the manner explained in Sec. II.

IV. Simulation Evaluation

In this section, we show that our proposal realizes reliable data collection even when additions or breakdowns of sensor nodes occur.

A. Simulation settings

We assume a 1,000 m $\times$ 1,000 m square region including the whole observed area and deploy N_s sensor nodes at uniformly random positions in the observed area. The communication range of a sensor node is represented by a circle with a radius of 100 m. Sensor nodes observe surrounding environmental phenomena and create a data packet that includes sensing data every one hour. Then, they forward the packet to one of the cluster heads. As an important assumption underlying reliable data collection, a retransmission algorithm can attain successful data transmission between two nodes and each node has sufficient memory not to drop any data.

In our simulation, randomly chosen N_b sensor nodes will fail, which causes other nodes to disconnect from the sub-network. Also, N_a sensor nodes will be added at random places in the observed area. Those failures and additions are taken at several pre-defined times.

A mobile sink, whose speed is set to 5 m/s, starts to move at 0 s and follows one of the proposed two mobility strategies. The mobility strategy of the mobile sink is being followed every one hour, and as the mobile sink finishes one strategy, it returns to the initial position to charge its battery. Here, the mobility strategy for learning sub-network positions is executed every LI hours.

We assume that the mobile sink can pass all the region of the observed area for evaluating basic performance. As an evaluation of reliable data collection, we calculate a data collection ratio (denoted by CR) every hour using (1).

$$CR = \frac{N_{CD}}{N_{ED}} \tag{1}$$

Here, N_{CD} and N_{ED} mean the number of collected and all generated data, respectively.

In our simulation, N_s, N_b, and N_a are set to 40, 5, and 10, respectively. Simulation time is set as 604,800 s (i.e., 1 week) and we performed 30 simulations with different sensor node positions for each LI ($LI = 1, 2, 3, 4, 5$).

B. Simulation results

Figure. 5 shows the transition of CR when LI is set to 1 and 5. In this figure, the Y-axis is the average of CR of 30 simulation trials. Even after node additions and failures occur, CR recovers from a temporary drop when LI is set to

 ISBN: 978-1-61208-391-9

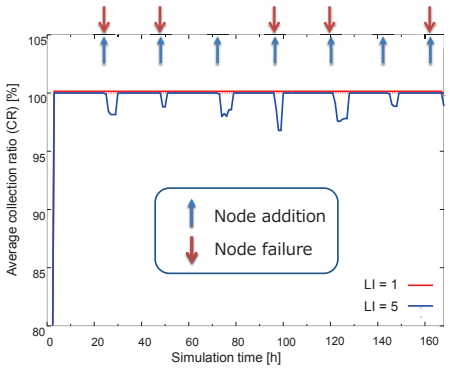


Figure 5. The transition of CR where $LI = 1$ and $LI = 5$

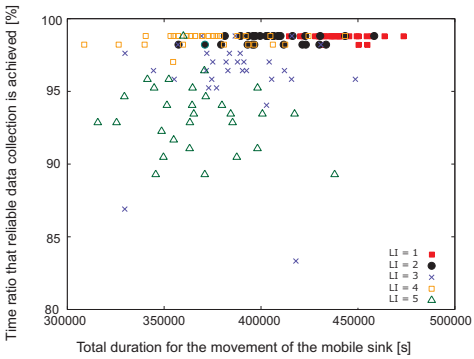


Figure 6. The relation between the reliability of data collection and the delay for mobility where $LI = 1, 2, 3, 4, 5$

5. This is because the mobile sink can learn the positions of new sub-networks every 5 hours. Thus, we find that reliable data collection can be attained by combining the two mobility strategies. Here, the average of CR remains 100% throughout the simulation when $LI = 1$. This is because the mobile sink always follows the strategy for learning the positions of all sub-networks.

Figure. 6 shows the relationship between the reliability and the efficiency of data collection. In this figure, the X-axis is the total time that the mobile sink moved except for waiting time at the initial position, and the Y-axis is the ratio of the time that the mobile sink can achieve 100% CR to the total simulation time.

The average durations for movement of the mobile sink are 438,428 s, 406,360 s, 385,653 s, 373,907 s, and 364,648 s, and the average time ratios that reliable data collection are achieved are 0.988, 0.986, 0.960, 0.986, and 0.934 for each $LI = 1, 2, 3, 4, 5$, respectively. The more frequent the mobile sink executes the mobility strategy for learning the sub-network positions, the more duration is required to acquire high reliability. Too frequent use of that strategy results in the redundant movements of the mobile sink in the case where few changes occur in the observed area. This trade-off relation between the reliability and the efficiency of data collection has to be managed with careful consideration of the frequency of environmental changes in the observed area.

V. Conclusion

In this paper, we present two mobility strategies for a mobile sink to realize reliable data collection. Our strategy can deal with a disconnection of a network and additional deployment of sensor nodes in an observed area. Through computer simulations, we demonstrate that reliable data collection is achieved by our proposal and show the trade-off between the reliability and the delay time for data collection. This trade-off can be adjusted by changing the frequency of the mobility strategy of the mobile sink for learning the positions of sub-networks. Our current interests are in the path planning among sub-networks, and in implementing our proposal in actual mobile nodes.

References

[1] H. Karl and A. Willing, *Protocols and architectures for wireless sensor networks*. Wiley, Oct. 2007.

[2] S. V. Deshpande and P. Devale, “Recent trends in using wireless sensor networks in industrial environment,” *International Journal of Computer Networking Wireless and Mobile Communication*, vol. 3, no. 3, pp. 11–20, Aug. 2013.

[3] J. Luo and J. Hubaux, “Joint sink mobility and routing to maximize the lifetime of wireless sensor networks: the case of constrained mobility,” *IEEE/ACM Transactions on Networking*, vol. 18, pp. 871–884, Jun. 2010.

[4] I. Chatzigiannakis, A. Kinalis, and S. Nikoletseas, “Sink mobility protocols for data collection in wireless sensor networks,” in *Proceedings of the 4th ACM International Workshop on Mobility Management and Wireless Access*, Oct. 2006, pp. 52–59.

[5] Z. Wang, S. Basagni, E. Melachrinoudis, and C. Petrioli, “Exploiting sink mobility for maximizing sensor networks lifetime,” in *Proceedings of the 38th Annual Hawaii International Conference on System Sciences*, Jan. 2005, pp. 287–295.

[6] F. Mourad, H. Chehade, H. Snoussi, F. Yalaoui, L. Amodeo, and C. Richard, “Controlled mobility sensor networks for target tracking using ant colony optimization,” *IEEE Transactions on Mobile Computing*, vol. 11, no. 8, pp. 1261–1273, Aug. 2012.

[7] R. Falcon, H. Liu, A. Nayak, and I. Stojmenovic, “Controlled straight mobility and energy-aware routing in robotic wireless sensor networks,” in *Proceedings of the 2012 IEEE 8th International Conference on Distributed Computing in Sensor Systems*, May 2012, pp. 150–157.

[8] S. Toyonaga, Y. Fujita, D. Kominami, and M. Murata, “Implementation of controlled sink mobility strategies with a gradient field in wireless sensor networks,” in *Proceedings of the 7th International Conference on Sensor Technologies and Applications*, Aug. 2013, pp. 27–32.

[9] Z. Liu, Q. Zheng, L. Xue, and X. Guan, “A distributed energy-efficient clustering algorithm with improved coverage in wireless sensor networks,” *Future Generation Computer Systems*, vol. 28, no. 5, pp. 780–790, May 2012.

[10] D. Kominami, M. Sugano, M. Murata, and T. Hatauchi, “Controlled and self-organized routing for large-scale wireless sensor networks,” *ACM Transactions on Sensor Networks*, vol. 10, p. 13, Nov. 2013.

 ISBN: 978-1-61208-391-9

Adaptation for Active Perception
Robustess from adaptation to context

Andreas Hofmann, Paul Robertson

Dynamic Object Language Labs Inc.
Lexington, Massachusetts, 02421
Email: {andreas, paulr}@dollabs.com

***Abstract*—Existing machine perception systems are brittle and inflexible, and therefore cannot adapt well to environment uncertainty. Natural perception always occurs in support of an activity that provides context. In our approach, we use context to adapt the perceptual aperatus providing robustness and resilience to noise among other benefits. Evidence supports the view that context driven adaptation occurs in natural perception systems including human vision. This *Active Perception* approach prioritizes the system's overall goals, so that perception and situation awareness are well integrated with actions to focus all efforts on these goals in an optimal manner. We use a Partially Observable Markov Decision Process (POMDP) framework, but do not attempt to compute a comprehensive control policy, as this is intractible for practical problems. Instead, we employ *Belief State Planning* to compute point solutions from an initial state to a goal state set. This approach automatically adapts the perception data flow processes, and generates action sequences for sensing operations that reduce uncertainty in the belief state, and ultimately achieve the goal state set. Our early results described in this paper demonstrate the feasibility of this approach using a restriced set of actions.**

Keywords: Active Perception, POMDP, Belief State Planning, Context, Adaptation.

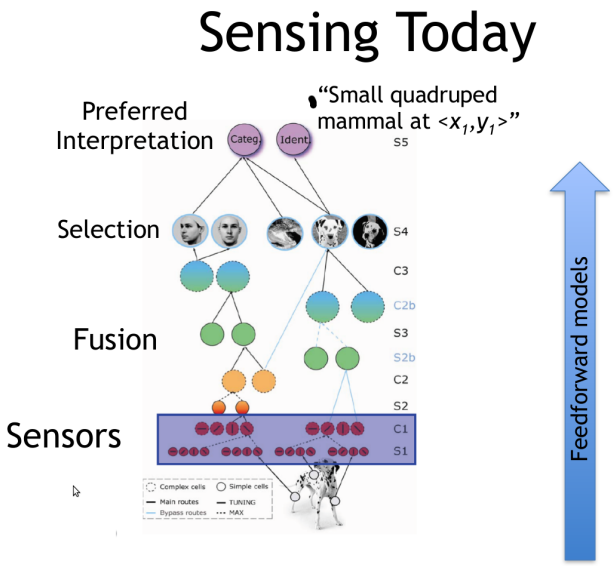


Figure 1. Sensing today is static and largely bottom-up.

I. Introduction

Natural perception systems such as the human visual system operate robustly in a wide variety of situations. Machine vision systems on the other hand have very little flexibility and tend to be very brittle in the face of environmental changes. The human visual system adapts with changing lighting conditions and with changing tasks. Adaptation is the key to this robustness. Machine vision systems work well when the environmental conditions and the task remain fixed but for robots to perceive their environment robustly for a variety of tasks and in ever changing conditions, requires an adaptive approach.

In traditional machine vision systems information flow is bottom up, and generally not guided by knowledge of higher level context and goals, as shown in Figure 1.

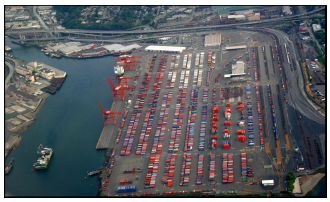
If higher level goals, context, or the environment change, the specific conditions for which the static configuration is intended may no longer hold. As a result, the static systems are prone to error because they cannot adapt to the new conditions. With few exceptions, such as [1], [2], they do not reason intelligently about dynamically changing goals, contexts, and conditions, and therefore, are not able to change to a more appropriate process flow configuration, or to change their parameter settings in an intelligent way.

In addition to their inflexibility, existing machine perception systems are often not well integrated into the autonomous and semi-autonomous systems to which they provide information. As a result, they are unaware of the autonomous system's overall goals, and therefore, cannot make intelligent observation prioritization decisions in support of these goals. In particular, it may not be necessary or useful for the perception system to be aware of every aspect of a situation, and it may be detrimental, due to resource contention and time limits, to the overall goal. Examples of autonomous systems operating in dynamic and uncertain environments, and that depend on intelligent perception, are shown in Figure 2. In a cyber attack scenario (sub-figure a), the overall goal of an autonomous system intended to provide security is to defend against the attack. Understanding the details of the attack is a subgoal. In a seaport operation scenario (sub-figure b), the overall goal of an autonomous control system is to operate the seaport safely

 ISBN: 978-1-61208-391-9

and efficiently. Understanding seaport activities is a subgoal. In an auto repair scenario (sub-figure c), the overall goal is to repair the problem. Diagnosing the problem is a subgoal. In all three cases, it may not be necessary to understand every detail of what is happening in order to accomplish the overall goal.

(a) Cyber attack.



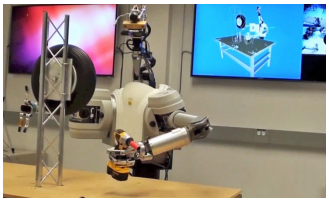
(b) Seaport.

(c) Auto repair.

Figure 2. Three example scenarios with dynamic environments.

Consider the automobile repair problem, or more specifically, the problem of changing a tire. This has been a "textbook" problem for Planning Domain Definition Language (PDDL) generative planning systems [3], and also a challenge problem in the Defense Advanced Research Projects Agency (DARPA) ARM project [4]. There are significant challenges in building an autonomous system that can perform this task entirely, or even one that would just assist with the task (Figure 3). Intelligent machine perception would be needed for both a fully autonomous system (sub-figure a), and a semi-autonomous advisory system (sub-figure b). The latter might include observation drones, and Google glasses [5] to guide a human user in the repair. Such a system would have to be able to determine the vehicle type (Figure 4), whether a tire is actually flat (Figure 5), and what an appropriate sequence of repair steps should be (Figure 6). It would have to be able to solve many subproblems, such as reliably finding a wheel in an image. The system would have to be able to work in many different environments, a great range of lighting conditions, and for a comprehensive set of vehicle types.

We address this challenging problem by using a more dynamic approach in which reasoning about context is used to actively and effectively allocate and focus sensing and



(a) Fully autonomous system using robot.

(b) Semi-autonomous advisory system.

Figure 3. Semi-autonomous and fully-autonomous systems.

(a) Truck

(b) SUV

(c) Sedan

Figure 4. Vehicle type, and ultimately, make and model, are useful things for the perception system to know (or to discover).

(a) Tire is flat.

(b) Tire is under-inflated.

(c) Tire is OK.

Figure 5. The perception system must be able to determine whether a tire is really flat, and which one it is.

(a) Wheel chock.

(b) Remove wheel.

Figure 6. Repair steps include stabilizing the car (a), getting the spare tire and tools, jacking up the car, removing lug nuts and wheel (b), and installing the new wheel.

action resources. This *Active Perception* approach prioritizes the system's overall goals, so that perception and situation awareness are well integrated with actions to focus all efforts on these goals in an optimal manner. Active perception draws on models to inform context-dependent tuning of sensors, direct sensors towards phenomena of greatest interest, to follow up initial alerts from cheap, inaccurate sensors with targeted use of expensive, accurate sensors, and to intelligently combine results from sensors with context information to produce increasingly accurate results (Figure 7). The model-based approach deploys sensors to build structured interpretations of situations to meet mission-centered decision making requirements. Actions are calibrated so that they are proportional to the likelihood of true detection and degree of threat.

We consider, in detail, a subproblem of the tire change problem: reliably finding the wheels of a vehicle in an image. We show how the use of top-down, model-based reasoning

 ISBN: 978-1-61208-391-9

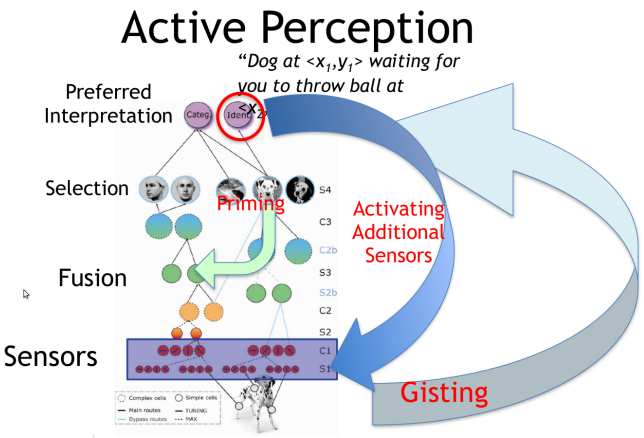


Figure 7. Active Perception uses model-based top-down reasoning, as well as bottom-up computation, to guide sensing actions.

can be used to coordinate the efforts of multiple perception algorithms, resulting in more robust, accurate performance than is achievable through use of individual algorithms operating in a bottom-up way. We also show how our model-based framework can be used to address the entire tire change problem.

The rest of this paper is organized as follows. In Section II, we present informal and formal statements of the problem to be solved. Section III covers related work, and Section IV introduces our approach. Sections V, VI, and VII describe the details of the approach, and Section VIII presents results. We conclude with a discussion in Section IX.

II. PROBLEM STATEMENT

Given one or more *agents* operating in an environment, and given that the agents do not directly know the state of the environment, or even, possibly, parts of their own state, and given a goal state for the environment and agents, the problem to be solved is to compute control actions for the agents such that the goal is achieved. In this case, an agent is a resource capable of changing the environment (and its own) state, by taking action. An agent could be a mobile ground robot, a sensing device, or one of many parallel vision processing algorithms running on a cluster, for example. Given that there is uncertainty in the environment state, because it cannot be measured directly, an agent must estimate this state as best as is possible based on (possibly noisy) observations. Similarly, part of the agent's state, such as the functional status of its components, may not be directly measurable, and must be estimated. Based on the agent's best estimate of the current environment state (and its own state), it should take actions that affect the state in a beneficial way (move the state towards the goal). The actions, themselves have some uncertainty; they do not always achieve the intended effect on the state. The agents must take both state estimate uncertainty, and action uncertainty into account when determining the best course of action. Actions can also have cost. The agents must balance the cost of actions against the reward of reduced uncertainty and progress towards the goal when deciding on actions.

This problem presents significant challenges. First, the overall state space, including environment and agent state, can be very large; there may be many state variables in the representation. Second, the state space is generally hybrid; it includes discrete variables, such as hypotheses for vehicle type, as well as continuous variables, such as position of a wheel. Third, significant parts of the state space may not be directly measurable, and must be estimated based on observations. The observations (usually sensor data) typically have some noise. Therefore, the estimates will have some uncertainty. As a result, the state variables themselves are generally not represented by a single value, but rather, by a probability distribution. These distributions must be updated as part of each state estimation cycle. Fourth, effect of some actions on state may have uncertainty. Fifth, the agents must take many considerations into account when deciding on actions: they must take into account the uncertainty of the state estimate, the uncertainty of the action effect on the state, the cost of the action, and the benefit of the action in terms of reducing uncertainty and making progress towards the goal.

The problem to be solved can be stated formally as follows. Suppose that the state space is represented by $S = \{S_e, S_a\}$, where S_e is the state space associated with the environment, and, S_a, that associated with the agent. Suppose, further, that the agent can perform a set of actions, A, and make a set of observations, O. A state transition model represents state evolution as a function of current state and action: $T : S \times A \times S \mapsto [0,1]$. An observation model represents likelihood of an observation as a function of action and current state: $\Omega : O \times A \times S \mapsto [0,1]$. A reward model represents reward associated with state evolution: $R : S \times A \times S \mapsto \mathbb{R}$. Given an initial state s_0 and goal states s_g, the problem to be solved is to compute an action sequence $a_0, \ldots a_n$ such that $s_n \in s_g$, and R is maximized. The agent bases its action decisions on its estimates of the state, which in turn, is based on the observations. The state estimates must be sufficiently accurate to support good action decisions. Note that this problem formulation, expressed in terms of a single agent, is easily extended to allow for multiple agents.

III. BACKGROUND AND RELATED WORK

A *Partially Observable Markov Decision Process* (POMDP) [6] is a useful framework for formulating problems for autonomous systems where there is uncertainty both in the sensing and in the results of actions taken. A POMDP is a tuple $\langle S, A, O, T, R, \Omega, \gamma \rangle$ where S is a (finite, discrete) set of states, A is a (finite) set of actions, O is a (finite) set of observations, $T : S \times A \times S \mapsto [0,1]$ is the transition model (conditional probabilities for state transitions), $R : S \times A \times S \mapsto \mathbb{R}$ is the reward function associated with the transition function, $\Omega : O \times A \times S \mapsto [0,1]$ is the observation model (conditional probabilities for state given observation and action), and γ is the discount factor on the reward. The *belief state* is a

 ISBN: 978-1-61208-391-9

probability distribution over the state variables, and is updated each control time increment using recursive predictor/corrector equations. First, the *a priori* belief state for the next time increment, $\bar{b}(s_{k+1})$, is based on the *a posteriori* belief state for current time increment.

$$\begin{aligned}\bar{b}(s_{k+1}) &= \sum_{s_k} Pr(s_{k+1}|s_k,a_k)\hat{b}(s_k) \\ &= \sum_{s_k} T(s_k,a_k,s_{k+1})\hat{b}(s_k)\end{aligned} \tag{1}$$

Next, the a posteriori belief state for the next time increment, $\hat{b}(s_{k+1})$, is based on the a priori belief state for next time increment.

$$\begin{aligned}\hat{b}(s_{k+1}) &= \alpha\, Pr(o_{k+1}|s_{k+1})\bar{b}(s_{k+1}) \\ &= \alpha\, \Omega(o_{k+1},s_{k+1})\bar{b}(s_{k+1})\end{aligned} \tag{2}$$

$$\alpha = \frac{1}{Pr(o_{k+1}|o_{1:k},a_{1:k})} \tag{3}$$

These equations work well given that the models are known, and given that the control policy for selecting an action based on current belief state is known. Unfortunately, computing these, particularly the control policy, is a challenging problem. Value iteration [7] is a technique that computes a comprehensive control policy, but it only works for very small problems. An alternative is to abandon computation of a comprehensive control policy, and instead, compute point solutions for a particular initial state and goal state set.

A promising technique for this is *Belief State Planning* [8], which is based on generative planner technology [9]. A key idea in this technique is the use of two basic actions for a robotic agent: move and look. A move action changes the state of the robot and/or environment; it may move the robot in the environment, for example. A look action is intended to improve the robot's situational awareness.

The move action is specified using a PDDL-like description language.

```
Move(lstart, ltarget)
  effect: BLoc(ltarget, eps)
  precondition: BLoc(lstart, moveRegress(eps))
  cost:  1
```

The variables lstart and ltarget denote the initial and final locations of the robot. The effect clause specifies conditions that will result from performing this operation, if the conditions in the precondition clause are true before it is executed. Cost of the action is specified in the cost clause.

The function BLoc(loc, eps) returns the belief that the robot is at location loc with probability at least (1 - eps). The moveRegress function determines the minimum confidence required in the location of the robot on the previous step, in order to guarantee confidence eps in its location at the resulting step:

$$moveRegress(eps) = \frac{eps - p_{fail}}{1 - p_{fail}} \tag{4}$$

where p_{fail} is the probability that the move action will fail.

The look action is specified as follows:

```
Look(ltarget)
  effect: BLoc(ltarget, eps)
  precondition: BLoc(lstart,
                     lookPosRegress(eps))
  cost:  1 - log(posObsProb
                     (lookPosRegress(eps)))
```

The lookPosRegress function takes a value eps and returns a value eps' such that, if the robot is believed with probability at least 1 eps' to be in the target location before a look operation is executed and the operation is successful in detecting ltarget, then it will be believed to be in that location with probability at least 1 eps afterwards.

$$lookPosRegress(eps) = \frac{eps(1-p_{fn})}{eps(1-p_{fn}) + p_{fp}(1-eps)} \tag{5}$$

where p_{fn} and p_{fp} are the false negative and false positive observation probabilities.

In terms of the POMDP belief state update, the move action corresponds to the predictor (1), and the look action corresponds to the corrector (2). It would also be possible to have actions that include both components. In our work, because we are currently focused on the observation aspect, our actions follow the approach of the look action, utilizing the observation model, but not a transition model. Also, the look action presented here implies independence of observations. It supports building confidence by repeatedly applying the look action (by staring). This makes sense in some contexts, but not in ones where a static image is being analyzed by an operator that does not change. Therefore, in our work, we cannot assume independence and handle the belief state update in an alternative manner.

For the actual observation algorithms, we make extensive use of two types of feature detection algorithms: *Speeded Up Robust Features* (SURF) [10], and *Hough Transforms* [11]. We utilize the Mathworks Computer Vision System Toolbox [12] implementation of these algorithms. Neither of these algorithms, used individually, is satisfactory for solving the wheel detection problem robustly. However, when used together, in an Active Perception framework, they beneficially reinforce each others hypotheses, allowing for more reliable performance.

 ISBN: 978-1-61208-391-9

IV. APPROACH

In order to achieve the Active Perception capabilities described in the Introduction, we define a process for Active Perception, as shown in Figure 8. The high-level *Meta-control* component maintains the system's current goals, and manages hypothesis formation and refinement. Hypotheses are maintained in a prioritize pool, and are used to guide *Sensor Fusion* and *Differential Diagnosis and Interpretation* components. These components provide the Meta-control component with an awareness of the current situation by strenghthening or weakening hypotheses, based on evidence from the sensors. The Meta-control component uses this awareness, combined with goals to be achieved, to task sensors in an optimal way, such that the goals will be achieved with greatest likelihood and efficiency.

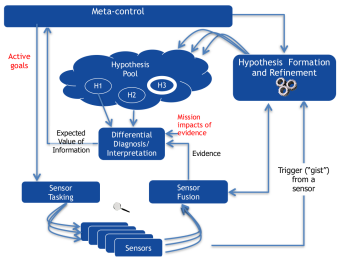


Figure 8. The Active Perception Process uses hypothesis formation and refinement to guide tasking of sensors.

In order to make this more concrete, it is useful to consider what a data flow diagram for the sensors and state estimation components should look like (if a computer vision expert were to solve the specific problem of designing an architecture for finding wheels in an image). Figure 9 shows a network of sensing components that collaborate to achieve the goal of finding wheels in an image with a high level of certainty. Sensing algorithms include feature matching, wheel location prediction, and Hough ellipse detectors. Each such sensing algorithm can use the current belief state as input, and can also adjust belief state as output. Sensing actions perform sensing operations by setting parameters for the sensing algorithms, running the algorithms, and interpreting the results. In particular, the sensing algorithms produce observations, which are used to update belief state.

This organization based on actions, observations, and belief state fits well with the POMDP formalism. Unfortunately, although POMDP's are a standard method of formulating this type of problem, solution of the POMDP can be challenging. In particular, value iteration approaches [7] that attempt to compute a comprehensive control policy are intractible for all but the simplest problems.

A promising alternative is to abandon the attempt to compute a comprehensive control policy, and instead, compute point solutions for a particular initial state and goal state set. This requires significant runtime computation, but is generally far more tractable than value iteration for this type of problem. This approach automatically synthesizes data flow processes such as the one in Figure 9, and generates action sequences for sensing operations that reduce uncertainty in the belief state, and ultimately achieve the goal state set.

We use three main sensing actions: SURF Match, SURF Match Other Wheel, and Hough Ellipse Match. Each of these actions has preconditions (requirements for current belief state), and post conditions (effects on belief state). SURF Match uses the SURF algorithm to perform a preliminary detection of a wheel in an image. SURF Match Other Wheel attempts to find the second wheel, also using SURF, given that the first wheel has been detected. This action also performs vehicle pose estimation, and refines the prediction of where the wheels are. Hough Ellipse Match uses either a Hough Circle or Hough Ellipse transform algorithm to refine the wheel location estimates.

The sequence of actions can be computed using a *generative planner*. A generative planner searches for action sequences that satisfy all the precondition and postcondition constraints, and that maximize the reward function. In order to be usable for this type of problem, two changes from traditional generative planning are necessary. First, traditional generative planners use deterministic operators, but the belief state representation is probabilistic. To solve this problem, we determinize the belief state operators used in the pre and post conditions of the actions [8]. This is accomplished by specifying thresholds for required belief. For example, a precondition for an action might require that a belief state variable be true with probability greater than 0.7. A postcondition might specify that a belief state variable should be true with probability of 0.8. The second change from traditional generative planning is in the overall framework in which planning and control interact. With traditional generative planning, a plan is generated, and is then executed in its entirety. This approach is not suitable for our type of problem, given the high levels of uncertainty, especially for the success of actions. Instead, we adopt a receding horizon control framework in which a plan is generated based on the current belief state, but only the first action of this plan is executed. After the action, the belief state is updated based on observations, and an entirely new plan is generated. The process then repeats with the first action from this new plan being executed. This approach is computationally intensive, but it ensures that all actions are based on the most current belief state.

To implement this receding horizon control framework, we use an architecture consisting of four main components, as shown in Figure 10. The executive manages the receding horizon control process. It maintains the current belief state, and the goal state set. At each control loop iteration, it passes these to the planner. The planner, if successful, returns a plan consisting of actions that transition the system from the current state to a goal state, if there are no disturbances. The executive takes the first action from the plan and executes it by dispatching the appropriate sensor operation. The sensor operation produces an observation, which is used to update the belief state. The process then repeats.

 ISBN: 978-1-61208-391-9

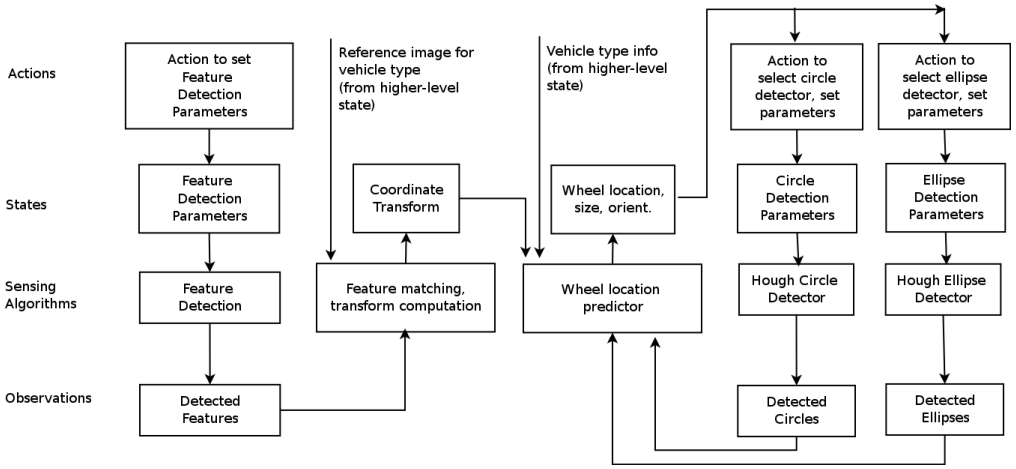


Figure 9. Data flow architecture for wheel finding components.

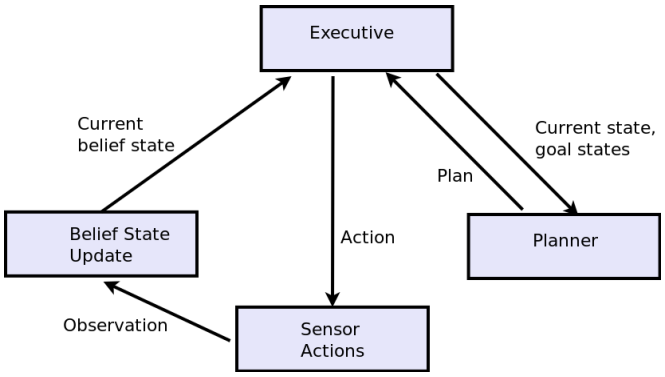


Figure 10. Receding Horizon Control Architecture for Active Perception

The following sections describe each of the components in this architecture in more detail.

V. EXECUTIVE

The Executive component implements the top-level receding horizon control loop, coordinating the activities of the planner and sensor components. Algorithm 1 shows pseudocode. The Executive accepts as input a goal state set, and an a priori belief state. It returns indicating success or failure in achieving a goal state.

The algorithm begins by initializing the belief state according to the a priori probabilities, and performing other initialization (Line 1). Belief state for a discrete state variable is represented as a *Probability Mass Function* (PMF) over the possible values of the probabilistic variable. Belief state for a continuous state variable is represented using a *Gaussian Probability Distribution Function* (Gaussian PDF) with a specified mean and variance. This could be extended to a representation using a mixture of Gaussians, in order to approximate more complex, non-Gaussian PDFs.

The receding horizon control loop begins at Line 5. The first step is to invoke the generative planner in order to determine the next action. A generative planner requires, as one of its inputs, the current state in a deterministic form. The belief state, however, is represented in a probabilistic form, so it first has to be converted to a deterministic form using MostLikelyState (Line 6). The input to the generative planner includes the determinized state, as well as the goal state set. This input is provided in the form of *domain* and *problem* PDDL files, as will be explained in more detail in Section VI. The Executive generates these files, and then executes the planner. The planner generates a result file containing a plan, which the Executive reads. The entire interaction of initializing the planner, running it, and retrieving the results is summarized in Line 7.

The planner may fail to generate a plan, in which case, the algorithm returns failure. If the planner is successful in generating a plan, the executive dispatches the first action (Line 11). The action (sensor operation) generates an observation. The belief state is updated based on this observation (Line 12).

At this point, the algorithm checks whether the goal has been achieved. (values of the state variables in the goal set have a sufficiently high belief). If not, the algorithm checks if the maximum number of allowed iterations have been exceeded (Line 15). If the maximum iterations haven't been exceeded, the algorithm begins a new control cycle, and the process repeats.

The Executive is implemented in Common Lisp. The Sensor Action and Belief State Update components are implemented in Matlab. The Executive communicates with these components using the C to Matlab API provided by Mathworks, combined with the Common Lisp to C foreign function interface provided by the Common Lisp implementation we use [13].

VI. PLANNER

The Planner component is implemented using *Fast Downward* [9], a state of the art generative planner that accepts problems formulated in the PDDL language [14]. A PDDL

 ISBN: 978-1-61208-391-9

Algorithm 2: Algorithm 1: Executive

Input: a-priori-belief-state-probabilities, goal-state-set
Output: goal-achieved?

```
                    /* Perform initialization.  */
1  belief-state ←
   InitializeBeliefState(a-priori-belief-state-probabilities);

2  goal-achieved? ← false;
3  max-iterations ← 1000;
4  iteration ← 0;
                          /* Begin control loop.  */
5  while not goal-achieved? do
6      current-state ← MostLikelyState(belief-state);
7      plan? ← GeneratePlan(current-state, goal-state-set);
8      if not plan? then
9          return ;
10     action ← First(plan?);
11     observation ← Dispatch(action);
12     belief-state ← UpdateBeliefState(observation);
13     goal-achieved?
       ← CheckGoalAchieved(belief-state, goal-state-set);
14     iteration ← iteration + 1;
15     if iteration > max-iterations then
16         return ;
```

problem formulation consists of a *domain* file, and a *problem* file. The domain file specifies types of actions that can be used across a domain of application such as logistics, manufacturing assembly, or in this case, finding wheels in an image. The domain file is fixed; it does not change for different problems within the domain. The problem file, on the other hand, contains problem-specific information such as initial and goal states. The Executive generates this file automatically. The following PDDL domain file fragment shows the definition of the SURF Match action in PDDL.

```
(:action SURF-match
  :parameters
    (?w ?p ?bsv)
  :precondition
    (and (belief-state-variable ?bsv)
         (pose ?p) (wheel ?w)
         (for-wheel ?w ?bsv)
         (at-pose ?p ?bsv)
         (at-belief-level
           belief-level-one ?bsv))
  :effect
    (and (at-belief-level
           belief-level-two ?bsv)
         (increase
           (total-cost)
           (feature-observation-cost ?p))))
```

The precondition clause specifies that the belief state variable value for a particular pose and wheel must be at belief level one in order for this operation to be tried. The effect clause specifies that the belief level increases to belief-level-two if the operation succeeds. The cost for the operation is also added to the total cost. The other sensor actions are specified in the PDDL domain file in a similar manner.

VII. SENSOR ACTIONS

We now describe in more detail the three sensor actions introduced previously: SURF Match, SURF Match Other Wheel, and Hough Ellipse Match.

A. SURF Match

The SURF Match action uses the SURF (Speeded Up Robust Features) algorithm [10] to attempt to identify wheels in the target image. SURF is inspired by SIFT (Scale-invariant Feature Transform), but is several times faster and can also be more robust against different image transformations. SURF is based on sums of 2D Haar wavelet responses and makes an efficient use of integral images.

The SURF Match action uses the SURF algorithm to attempt to match a wheel in a reference image with a wheel in the target image. The SURF algorithm is scale invariant, but is somewhat sensitive to large changes in orientation. Therefore, multiple reference images are used, including ones for different orientations. The orientations in the reference images (see Figure 11) correspond to the orientations of the discrete belief state variables.

(a) +2 (b) +1 (c) 0 (d) -1 (e) -2

Figure 11. Wheel reference images, corresponding to different orientations.

The SURF Match action is not highly reliable; it can miss detecting a wheel, especially if the target image is noisy, it can falsely detect objects that are not wheels. Also, due to the symmetry of wheel images, the SURF Match algorithm does not provide a very accurate estimate of the pose (position and orientation) of the wheel in the target image. Thus, the SURF Match action is used to attempt to achieve a rough initial match, the goal being to move from low to medium confidence estimates, and set the stage for the use of the other sensor actions to improve the estimates.

We used the functions detectSURFFeatures, extractFeatures, matchFeatures, and estimateGeometricTransform from the Mathworks Computer Vision System Toolbox [12] to implement the SURF Match action. Parameters used for these functions are shown in Tables I, II, and III.

 ISBN: 978-1-61208-391-9

TABLE I. PARAMETERS FOR detectSURFFeatures

MetricThreshold	NumOctaves	NumScaleLevels
100	5	5

TABLE II. PARAMETERS FOR matchFeatures

MatchThreshold	MaxRatio
30.0	0.7

In our implementation of this sensor action, we used fixed parameters for these functions, but this could be extended to allow for adaptively adjusting parameters within the active perception framework. For example, the MaxDistance parameter of the estimateGeometricTransformParameters can be beneficially tuned to improve performance. Besides estimating the transform, this function removes outlier matches. If the MaxDistance parameter is too small, it will remove too many points that are not outliers. Conversely, if the value is too large, too many outliers remain. We have set the value to 20.0 (up from its default setting of 1.5), and found this worked well in the tests we performed. However, there may be conditions that we haven't tested under which a lower setting is better.

B. SURF Match Other Wheel

The SURF Match Other Wheel sensor action is similar to the SURF Match action, but assumes that one wheel position estimate already exists (from a previous SURF Match action). It uses this information to try to find the other wheel. This action uses a number of gists. In this case, the gists come from previous observations or assumptions, external to the wheel finder system. In particular, it is assumed that the action is observing the left side of the car, and that the car is on level terrain.

The SURF Match Other Wheel action uses the existing wheel position estimate to crop out that part of the image. This forces the SURF algorithm to look elsewhere for the other wheel.

Additionally, if the SURF Match Other Wheel action is successful in finding the other wheel, then it uses the position estimates of both wheels to estimate the pose of the overall car. Note that the position estimates of the wheel resulting from the SURF algorithm are image position estimates. The sensor action uses projective geometry, combined with a number of additional gists, to estimate the positions of each wheel in the world coordinate frame, given the image position estimates. These gists are: 1) the height of the camera; 2) the focal length of the camera; and 3) the size of the wheel. All of these are reasonable gists; a ground robot or UAV would know the focal length of its camera, as well as its height. Given previous gists for vehicle type, the size of the wheel can be determined from the vehicle type's spec data.

Once the position estimates of each wheel in the world coordinate frame are known, simple trigonometry is used to determine orientation of the car, particularly, its yaw (rotation about the vertical axis). This estimate is the continuous counterpart to the discrete belief state variables for orientation. The continuous and discrete variables inform and reinforce each other as part of the belief state update mechanism.

TABLE III. PARAMETERS FOR estimateGeometricTransform

MaxNumTrials	MaxDistance
20000	20.0

C. Hough Ellipse Match

The Hough Ellipse Match sensor action uses Hough transforms [11] to determine wheel position in an image with a high degree of accuracy. This action is computationally expensive, but has the potential to give very accurate estimates, when supplied with good parameters. Thus, this action is used after the other, less expensive sensor actions have developed a strong hypothesis about where a wheel might be, and what it's orientation is likely to be.

The computational expense of the Hough transform algorithm rises as the number of parameters increases. The variation of the Hough transform for detecting lines is the cheapest; it requires only two parameters. The circle variation is more expensive since a circle is described by three parameters (the x, y position of the center, and the radius). The ellipse variation is still more expensive since an ellipse is described by six parameters. For this reason, the Hough Ellipse Match sensor action first checks whether estimated orientation (yaw angle) of the car is small, indicating that the car side is facing the camera directly. In this case, the sensor action employs the circle variation of the Hough transform, as a special case of an ellipse, since the circle variation is faster. Even so, this algorithm only works well if good bounds can be specified for the parameters (circle center and radius).

For the more general case, we use an efficient Hough Ellipse algorithm [15] that requires good bounds for major axis length, aspect ratio, and rotation angle. This algorithm can take a very long time (30 minutes or more in some cases, depending on image size), and is very sensitive to the parameter bounds. In particular, it can easily incorrectly match non-wheel objects as ellipses if its search area and parameters are not tightly constrained. For this reason, we crop the image based on the current estimated wheel position in the image, to reduce the area where the algorithm has to search. Additionally, we use orientation and scale information computed by the other sensor actions to constrain the major axis, aspect ratio, and rotation parameters.

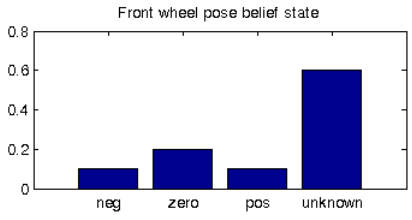
VIII. RESULTS

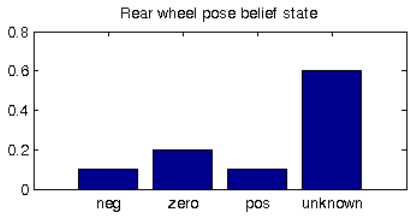
A. Results

In this section, we provide an end-to-end example run showing adaptation to the sensors as knowledge is learned from the image.

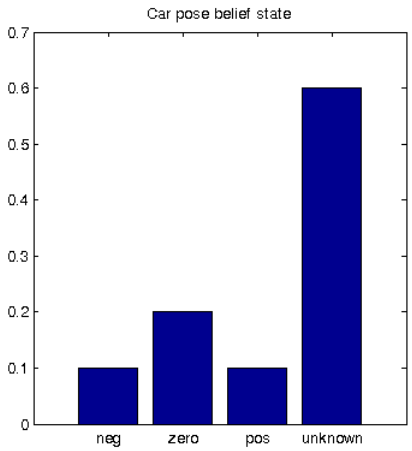
The a priori belief state is shown in Figure 12. This indicates that the poses are largely unknown, with a slight bias to the zero pose.

 ISBN: 978-1-61208-391-9





(a) Wheel



(b) Car

Figure 12. Wheel and car pose, a priori belief state.

The first control step iteration, based on this belief state, yields the following plan from the PDDL planner.

```
1. SURF Match (front-wheel pose-zero)
2. SURF Match Other Wheel (front-wheel pose-zero)
3. Hough Ellipse Match (rear-wheel pose-zero)
```

The Executive performs the first of these actions, yielding a successful match, as shown in Figure 13.

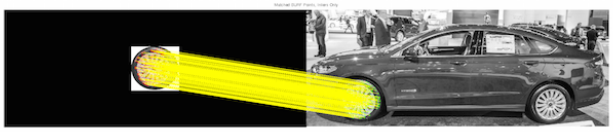


Figure 13. Successful SURF match to front wheel.

Based on this, wheel pose estimates are updated as shown in Figure 14; the hypothesis for zero pose for the front wheel has been strengthened over the a priori estimate shown in Figure 12.

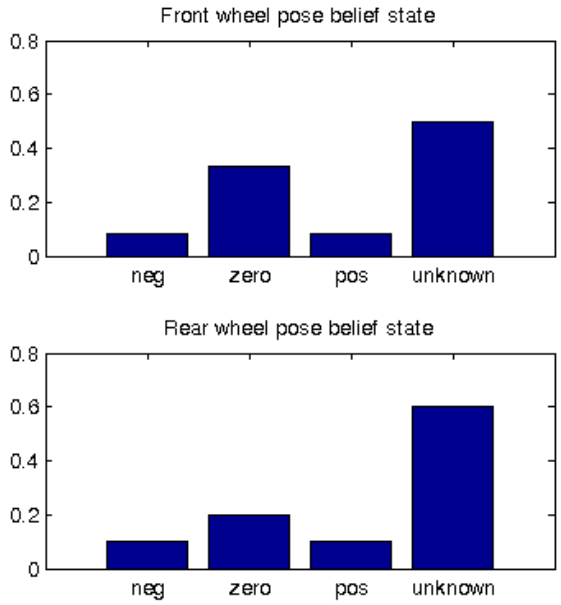


Figure 14. Wheel pose belief state after initial SURF match.

The second control step iteration, based on this updated belief state, yields the following plan from the PDDL planner.

```
1. SURF Match Other Wheel (front-wheel pose-zero)
2. Hough Ellipse Match (rear-wheel pose-zero)
```

The Executive performs the first of these actions, yielding a successful match, as shown in Figure 15. Note that the front wheel has been cropped out to focus the sensing action on the other (rear) wheel.

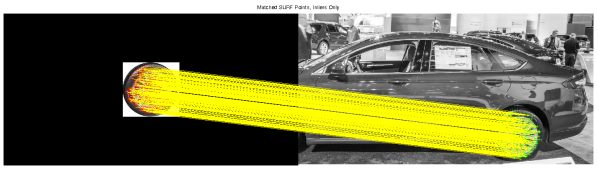


Figure 15. Successful SURF match to other (rear) wheel.

Based on this, wheel and car pose estimates are updated (see Figure 16), the hypotheses for zero pose for the wheels and car have been strengthened.

The third control step iteration, based on this updated belief state, yields the following plan from the PDDL planner.

```
1. Hough Ellipse Match (rear-wheel pose-zero)
```

The Executive performs the first of these actions, yielding a successful match, as shown in Figure 17.

The final wheel belief state estimates are shown in Figure 18.

 ISBN: 978-1-61208-391-9

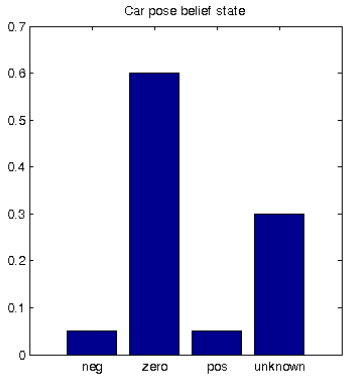


Figure 16. Car pose belief state, after SURF Match Other Wheel action.

Figure 17. Successful Hough ellipse match to rear wheel (match highlighted in green).

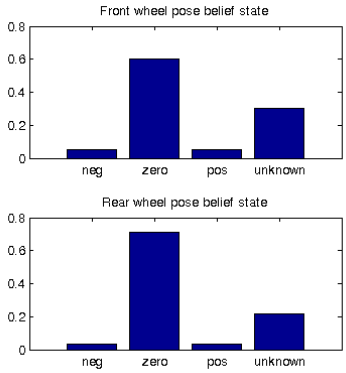


Figure 18. Final wheel pose belief state.

IX. DISCUSSION

The focus of our efforts thus far has been on the sub-problem of finding a wheel in an image. This has led to an emphasis on "look" actions, but no incorporation of "move" actions (actions that change the state of the environment). As stated previously, this is only part of the larger problem of autonomously or semi-autonomously repairing a car tire. We believe that the approach we have developed is well suited for extension to the larger problem, and problems like it. In particular, it is well suited to incorporating move as well as look actions, with the generative planning component intelligently combining both types of actions. This would allow for testing with more general kinds of problems, where the goal is more than purely a perception goal, but rather, involves changing the environment state to achieve an environment goal.

X. CONCLUSIONS

We have demonstrated the approach of using a PDDL planner with receding horzon planning to produce a context driven feature extraction capability. We are working now to integrate this capability within a framework that encompasses real actions that lead to a problem solving activity being performed (changing a tire) and in which many more observation actions are possible. To handle the increased state space size we are moving to a monte-carlo planner.

ACKNOWLEDGEMENTS

This research was developed with funding from DARPA. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

REFERENCES

[1] G. Hoelzl, M. Kurz, and A. Ferscha, "Goal processing and semantic matchmaking in opportunistic activity and context recognition systems," in The 9th International Conference on Autonomic and Autonomous Systems (ICAS2013), March 24 - 29, Lisbon, Portugal, textbfBest Paper Award, March 2013, pp. 33–39.

[2] ——, "Goal oriented recognition of composed activities for reliable and adaptable intelligence systems," Journal of Ambient Intelligence and Humanized Computing (JAIHC), vol. 5, no. 3, July 2013, pp. 357–368.

[3] M. P. Fourman, "Propositional planning," in Proceedings of AIPS-00 Workshop on Model-Theoretic Approaches to Planning, 2000, pp. 10–17.

[4] P. Hebert et al., "Combined shape, appearance and silhouette for simultaneous manipulator and object tracking," in Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012, pp. 2405–2412.

[5] N. Bilton, "Behind the google goggles, virtual reality," New York Times, vol. 22, 2012.

[6] G. E. Monahan, "State of the arta survey of partially observable markov decision processes: Theory, models, and algorithms," Management Science, vol. 28, no. 1, 1982, pp. 1–16.

[7] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable markov decision processes," arXiv preprint arXiv:1106.0251, 2011.

[8] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," The International Journal of Robotics Research, 2013, p. 0278364913484072.

[9] M. Helmert, "The fast downward planning system." J. Artif. Intell. Res.(JAIR), vol. 26, 2006, pp. 191–246.

[10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," Computer vision and image understanding, vol. 110, no. 3, 2008, pp. 346–359.

[11] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," Communications of the ACM, vol. 15, no. 1, 1972, pp. 11–15.

[12] P. Corke, "Matlab toolboxes: robotics and vision for students and teachers," IEEE Robotics & automation magazine, vol. 14, no. 4, 2007, pp. 16–17.

[13] H. Sexton, "Foreign functions and common lisp," ACM SIGPLAN Lisp Pointers, vol. 1, no. 5, 1987, pp. 11–23.

[14] D. McDermott et al., "Pddl-the planning domain definition language," 1998.

[15] M. Simonovsky, "Ellipse detection using 1d hough transform," http://www.mathworks.com/matlabcentral/fileexchange/33970-ellipse-detection-using-1d-hough-transform retrieved: January, 2015.

 ISBN: 978-1-61208-391-9

Multi-Agent Model for Leader Identification in Platoon System

Baudouin Dafflon

DISP-LAB
Université Lumière Lyon 2, Lyon, France
Email: baudouin.dafflon@univ-lyon2.fr

Franck Gechter

IRTES-SET
UTBM, Belfort, France
Email: franck.gechter@utbm.fr

***Abstract*—In the past decades, the research on autonomous vehicles and transportation systems has performed a great breakthrough. Among the emergent new transportation modes, the development of platoon control solutions seems to be very promising in terms of environmental impact and traffic jam aspects. The two main approaches generally encountered in literature deal with either a global point of view or a local one. In the local approaches, the follower vehicle perceives its environment, identifies a leader and applies a function to calculate a command. This paper deals with the identification task for a local platoon control system. This identification task is made using the reactive multi-agent paradigm. In the proposed system, the identification task can be defined as a selection of one pattern from a set following several criteria. These patterns are emergent structures made of agents which aggregate on specific areas of their environment depending on their perception and their interactions. The agent environment is built using data collected by sensors. The sensors raw data are processed so as to be integrated into agent environment. The association between one physical sensor and a suitable processing algorithm is called an abstract sensor. The paper presents in detail the proposal and its applications in simulated and real environments.**

Keywords–multi-agent; platoon system; leader identification

I. INTRODUCTION

In the past decades, the research on autonomous vehicles and transportation systems has achieved a great breakthrough with a widespread use of powerful embedded systems which includes multiple sensors and top level computational resources. In parallel with the extension of the autonomous abilities of individual vehicles, new transportation systems emerged. Among them, one can cite the development of platoon control solution aimed at helping the driver in his task while bringing some interesting properties in terms of environmental impact and traffic jam aspects.

A platoon can then be defined as a set of vehicles, evolving together without material link while keeping a given geometric configuration. In literature, two families of approaches are widely developed, they are classified according to a global or local reference frame. Global approaches propose to locate vehicles in a common reference frame shared by all the vehicles of the convoy. This requires precise localisation algorithms and efficient communication exchanges between vehicles. By contrast, local approaches are more reactive and they focus on local perception abilities. In these approaches, each vehicle perceives its environment, identifies a leader and applies a function to calculate a command. Thus, the local approaches rely mainly on an identification function aimed at finding the right vehicle to follow. Several solutions can be used to perform this identification task. For instance, one can cite the use of specific visual beacons [1], the use of Fuzzy logic algorithms [2] or the use of arithmetic solutions [3]. The main drawback of these solutions is a low robustness to interference or to sensor perturbations and the lack of adaptation ability so as to make them able to face with changes in scene configuration. To overcome these limitations, we propose, in this paper, a solution based on the reactive multi-agent paradigm. This solution uses the adaptive skills and the self-organization properties of multi-agent systems so as to provide robustness and adaptability to the identification system.

In recent years, multi-agent systems have been widely used to solve dynamic problems such as dynamical obstacles avoidance [4], localization and tracking, robot coordination etc. It has been also demonstrated that reactive multi-agent system approaches are efficient for tackling complex problems such as autonomous parking control [5], cooperation of situated agents/robots [6], data fusion and problem/game-solving [7].

The goal of this paper is to present an approach for a vehicle identification problem in a platoon context. The proposed approach is based on the application of reactive multi-agent systems. The identification problem can be defined to be the activity of selecting pertinent elements from a set of data using considerations on their shape, structure, dynamic, etc. Applied to vehicle identification in local platoon context, the identification is aimed at selecting, among all the objects detected in the vehicle perception range, one vehicle that can be considered as a leader for the platoon task. Thus, the result of an identification is a set of objects containing the leader and obstacles. In our multi-agent approach, this set is defined by the observation and the study of the components and the properties of the multi-agent system. The agents we developed are immaterial and evolve in an virtual environment which is an abstraction of vehicle perception range. Agent-to-agent and agent-to-environment interactions are proposed to produce an agent dissemination into the virtual environment the spatial configuration of which is led by the data furnished by vehicle sensors. This emergent structure, which represents the global system state, is analysed by indicators which allow to differentiate a potential leader from obstacles and other vehicles.

The paper is structured as follows : Section II draws a state of the art of the platoon issue through a description of the past and current international projects on the subject. Section III describes the multi-agent model used. Section IV exposes simulations and experimentations of identification

 ISBN: 978-1-61208-391-9

model associated to the platoon function and Section V draws a conclusion of this work and provides clues for future research works.

II. BACKGROUND

The vehicle detection for the local platoon issue has been widely developed through a huge number of international projects. The first project (*PATH 1992-2003*) that deals with the platoon control was based on the use of radar sensors [8], which enable the detection system to measure with high accuracy the leader position. The drawback of this solution was that the following vehicle detects also obstacles and cannot separate them from the leader. Then, the *DEMO 2000* project proposes a system for the detection of obstacles in order to recognize and localize obstacles [9]. Effective on a straight road, the performance of this system is limited with curved trajectories and in an open environment. The *CHAUFFEUR* project [10] used an on-board image processing system to determine the relative position of the preceding vehicle. This system depends on the detection of the infrared light (IR) emitters attached to the back of the preceding vehicle. This solution is efficient while dealing with homogeneous platoons. In the case of trucks and regular cars being together in the same platoon, the size of IR emitters becomes a problem (some of them are too large for being attached to a small car). Moreover, the use of such artificial beacons increases the cost of the solution and the reliability of the systems depends strongly on the light conditions. As opposed to this, the *KONVOI* project [11] developed a driver assistance system (ADAS) which controls the longitudinal and lateral adjustment. This system is based on laser range finder and radar sensors. However, both devices are not usable in unstructured environments, because they are based on the retrieval of infrastructure information, such as lane markings or Geographical Information Systems as references for the longitudinal and lateral control.

III. MULTI-AGENT MODEL FOR IDENTIFICATION

A. Global overview

As previously explained, the proposed approach is based on the application of reactive multi-agent system [12] [13]. A virtual environment is built up using processed data collected by the embedded sensors of the follower vehicle. The association of one physical sensor and of a suitable processing algorithm is named abstract sensor. Agents are then spread in this environment. Depending on their perception and their interactions, agents aggregate on specific areas of their environment. This aggregation phenomenon leads to emergent structures made of agents which are considered as patterns. The task of identification can then be defined as a selection of one pattern from a set following several criteria.

The process can then be summarized by the following main steps (Figure 1):

1) Data are collected by abstract sensors.
2) Data are projected in 2D-space which corresponds to the environment of the agents ; this space is an abstraction of the state space of the problem.
3) A population of agents interacts with the projected data in this space using a set of interaction inspired by physics. The result of these interactions are structures or patterns which emerge into the environment.
4) A static and dynamic study allows then to identify the leader vehicle and the obstacles present in the vehicle's neighbourhood.

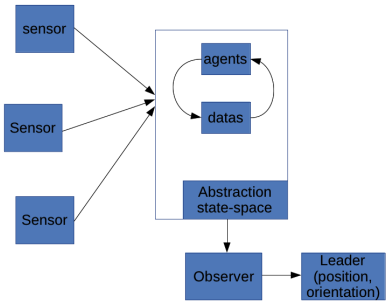


Figure 1. Global overview

B. multi-agent model

This section is aimed at giving a detailed description of the proposed Reactive Multi-Agent System (RMAS). The proposed approach puts the environment in the center of the problem-solving process. The environment corresponds to the place where the problem and its constraints are specified and presented to the perception of the agents. Then, interactions are defined in order to take into account the dynamics of the problem and its representation in the environment. These elements lead to emergent structures which are then analysed so as to extract the best solution to the problem. In the context of leader vehicle identification, the emergent structure is interpreted as a pattern with a specific geometrical shape and a particular behaviour.

1) Environment: As explained before, agents environment is the corner stone of the approach. It links vehicle's world and the identification mechanism. It is composed of entities associated to objects perceived by abstract sensors. An abstract sensor unit is composed of a sensor (software or hardware) and function of pre-processing. Objects are projected into environment as point or cloud of points.

2) Agents: The role of agents is to cover the environment, to locate and to track projected data. Two operations are considered:

- Grouping agents on the pertinent information.
- Interpreting the features of emergent structures.

In order to do this, two populations of agents are created, **label** agents and **delegate** agents.

The aim of **label** agents is to cluster, to follow and to isolate data of the virtual environment. Label agents have an internal state defined by one Label and one constraint.

- **The label**, denoted L_t^i for an agent A_i at t time, determines, for agent A_i, the membership to one group of agents according to spatial proximity. The value of the label is a natural number and is defined as follows:s

$$L_{t+1}^i(v) = \begin{cases} 0 & \text{if } \epsilon_t = \emptyset \\ \text{Rand(0,255)} & \text{if } \alpha_t = \emptyset \wedge \epsilon_t \neq \emptyset \\ min(L_t^i, L_t^j) & \text{if } \forall j \in \alpha_t \\ & (L_t^i \neq L_t^j \wedge L_t^i \neq 0) \end{cases} \quad (1)$$

where

- ϵ_t is a set of information
- α_t is a set of agents

- **The constraint** is a numerical value which represents the spatial organization of agents in the group. Considering an agent A_i, and its neighbourhood composed by the nearest two agents A_j and A_k, having the same label, the angle described by $\vec{ij}$ and $\vec{ik}$ defines the constraint. The higher the angle value is, the lower is the stress. This allows to define a pattern profile that can be used to identify the emergent structures.

The aim of **delegate** agents, is to detect and to locate groups of label agents having the same label value. Their internal state is defined by one satisfaction value and one vector named normal vector.

- The **satisfaction** is achieved when the delegate agent is near a group of label agents. The satisfaction value is progressive. Before reaching a threshold which locks its state, the delegate agent must satisfy constraints such as proximity, moving and loyalty to the group of label agents. A delegate agent who oscillates between two labels cannot be considered to be satisfied.
  The satisfaction value S^i_{t+1} increases in time, and can be calculated for an agent A_i at time t by:

$$S^i_{t+1} = \frac{1}{|\lambda|}\sum_{j\in\lambda}(1 - \frac{||\vec{r_t}\vec{j_t}||}{\pi}) + S^i_t \qquad (2)$$

where

λ = label of the group
$\vec{r_t}$ = position of the delegate agent
$\vec{j_t}$ = position of a label agent A_j

- The **normal vector** can be defined as a unit vector, collinear to the average motion vector of the agents pattern. It is computed when the delegate agent has reached a satisfaction threshold. The normal vector, noted $\vec{N^i}$ is defined as the mean of normal vectors of all the label agents composing the pattern using the following equation:

$$\vec{N^i} = \frac{1}{|\lambda|}\sum_{j\in\lambda}\vec{j_{dir}} \qquad (3)$$

The combination of the values of the internal states allows to cluster and to identify groups of agents.

3) Interactions: So as to reach their goals (relevant patterns of data), agents are evolving in their environment according to their perceptions and their interactions. The moves of agents are computed according to interactions inspired by classical physics. Label agents can thus be compared to particles in a force field influenced by their neighbourhood. Two types of interactions are used:

- **Agent-agent interaction** : agents are repulsing each other according to their nature (i.e. label agent repulses other label agents and delegate agent repulses other delegate agents). This ensures a homogeneous dispersion in the environment. The repulsion is computed following a classical gravitational Newton law in $1/r^2$. The force is given by :

$$\vec{Fr_{ij}} = \alpha m_i m_j \frac{\vec{A_iA_j}}{\|\vec{A_iA_j}\|^3} \qquad (4)$$

where

- m_i is the mass of agent i.
- A_i is the position of agent i.
- $\alpha$is a coefficient taking into account the state of the agent (if it is locked or not for a label agent, if it is satisfied or not for a delegate agent) and the gravitational constant of the environment which is set empirically.

- **label agent-delegate agent**: an attraction force is applied between agent and environment's items. Thus, label agents are attracted by the data which corresponds to the presence of perceived objects. Moreover, delegate agents are attracted by label agents by using the same kind of force. The mathematical formulation of the attraction force is given by :

$$\vec{Fa_{ic}} = \beta_g m_i m_c \frac{\vec{A_iC}}{\|\vec{A_iC}\|^3} \qquad (5)$$

where

- m_i is the mass of agent i.
- A_i is the position of agent i.
- m_c is mass of the center of attraction.
- C is position of the center of attraction.
- β_g is a coefficient taking into account the state of agents and environment properties.

Interactions are applied following Newton's law of motion. The system calculates at each time step, the position, the velocity and the acceleration of each agent. The acceleration is calculated as follows:

$$\begin{cases} \sum\overrightarrow{F_i} = m.\overrightarrow{\gamma_i} \\ \overrightarrow{\gamma_i} = \frac{1}{m}.\sum\overrightarrow{F_i} \\ \overrightarrow{\gamma_i} = \frac{1}{m}.(\overrightarrow{Fa} + \overrightarrow{Fr}) \end{cases} \qquad (6)$$

Forces can be generalised :$\overrightarrow{Fa}$ et $\overrightarrow{Fr}$ to all elements present in the perception field of agent A_i :

$$\begin{cases} Fr_i^X = \sum_{i\neq j}\left(m_i.m_j\frac{(x_j-x_i)}{((y_j-y_i)^2+(x_j-x_i)^2)^{\frac{3}{2}}}\right) \\ Fr_i^Y = \sum_{i\neq j}\left(m_i.m_j\frac{(y_j-y_i)}{((y_j-y_i)^2+(x_j-x_i)^2)^{\frac{3}{2}}}\right) \end{cases} \qquad (7)$$

and

$$\begin{cases} Fa_i^X = \alpha.m_i.m_c.(x_i - x_c) \\ Fa_i^Y = \alpha.m_i.m_c.(y_i - y_c) \end{cases} \qquad (8)$$

 ISBN: 978-1-61208-391-9

For agent A_i :

$$\left\{ \vec{X}_i(t) = \vec{X}_i(t-1) + \left(\vec{V}_i(t-1)\delta t + \frac{(\delta t)2}{2m} \left(\overrightarrow{Fr}_i + \overrightarrow{Fa}_i \right) \right) \right. \quad (9)$$

where

- $\vec{X}_i(t)$ is the agent position.
- $\vec{V}_i(t)$ is the agent velocity

On a computer implementation point of view, the agents behaviours are ruled by a scheduler. This scheduler calls, at each time step, a function aimed at computing the forces to be applied and the future acceleration, speed and position of each agent. The result of this scheduler behaviour is a movement of all agents in the environment involving emergent structures which can then be studied by an external software module detailed in next paragraph.

C. Observation and identification

The external software module is able to retrieve, at any moment, the position and speed of the agents. The emergent structures, that appear in the environment, are patterns sharing the same label and the same delegate. To achieve the identification of objects, the constraint values of each label agent is used. These values are collected by the delegate agent of the group and form a vector which represents the profile of the detected object. This vector can be compared to a application dependant database previously defined. The database contains a set of profiles defined by an expert and takes into account the characteristics of the problem. In the context of platoon system, the useful information is classified into three types: leader, vehicle and obstacle. Among the detected vehicles, one of them must be chosen to be the leader. To do this, a multi-criteria comparison between delegate agents characteristics is performed. The criteria taken into account are: the life time of the delegate agent in the environment, the distance between the delegate agent and the follower vehicle which needs to determine its leader, the angle between the delegate and the follower vehicle, the relative speed of the delegate in the environment and the fact that the delegate already represents the current local leader.

These criteria are expressed in the form of a radar chart (Figure 2) and the selection is made by comparison of the obtained surfaces. The delegate agent who has the largest surface area is considered to be the local leader, other delegate agents are considered as obstacles.

IV. Simulations and Experimentations

In this section, we present both results obtained in simulation and experimentation with real vehicles. So as to better compare both, the same protocol, scenarios, and metrics have been used.

A. Protocols

The testing protocol follows a classical workflow: data acquisition in a track involving all desired scenario, offline processing of data to produce classification, comparison between automatic classification and human classification.

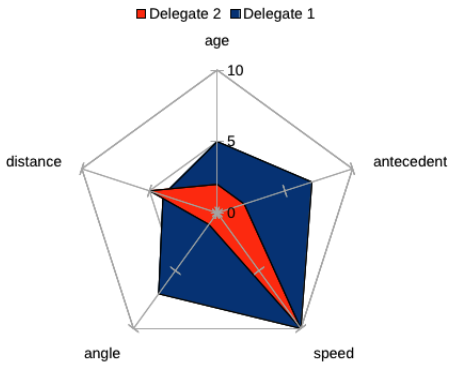


Figure 2. Delegate agents representation

1) Scenario: The scenario definition is one of the most important key points in simulations and experimentation. Tests have been made in the Technome site of Belfort. Technome site is an industrial/commercial activity area where pedestrians, parked and moving car, urban furniture can be found. Moreover, we have to the opportunity to use a 3D and geo-localised model of this area, that allows to proceed tests with the same scenario in simulation and in experimentation. The path selected for testing consists of several straight lines, a roundabout and various curves. It is surrounded by buildings and frequently crossed by pedestrians and cars. This path is repeated several times in order to obtain a sufficient amount of data.

2) Metrics: The metrics corresponds to the way the results are evaluated. Several metrics can be chosen such as the F-measure for example. In this case, we have chosen to concentrate on the application field. The metrics used are thus the classification rate between 2 or 3 classes (building, car, urban furniture, etc.), the false positive rate, the frequency and the duration of mistakes.

3) Tools:

- **VIVUS Simulator:** to assess the quality of our approach, realistic simulations have been done using VIVUS simulator [14], a vehicle simulator developed by the IRTES-SET laboratory. VIVUS is based on PhysX for real physical behaviour, and Unity3D for good 3D performance. This software can simulates the behaviours of each vehicle on several levels such as perceptions with laser range finder or cameras, physical reaction between elements (wheels, car's parts, etc.), etc. Physical reactions are computed using the same physical laws as in the real world (collision, gravity, etc.) and taking into account the properties of the environment (friction with road, materials of ground and walls, weather conditions, etc.). VIVUS has already been used to test various intelligent vehicle algorithms such as linear platoon control [13], obstacle avoidance and driving assistance [12], and intelligent crossroads simulations in [15].

- **SeT-Car platform** : the Systems and Transportation laboratory has got some electrical cars equipped for perception and autonomous navigation. The vehicle

used for these experimentations is equipped with various sensor such as a Real Time Kinematic GPS (differential GPS), video cameras, gyroscope, laser range finder, etc. In these simulations, only the laser range finder has been used. Its characteristics are the following: 180 degrees of aperture, 80 meters of range and 1 degree of resolution.

- **Janus** is a multi-agent platform that was specifically designed to deal with the implementation and deployment of holonic and multi-agent systems. It is based on an organizational approach and its key focus is that it supports the implementation of the concepts of role and organization as first-class entities. This consideration has a significant impact on agent implementation and allows an agent to easily and dynamically change its behaviour.

B. Simulation results

The goal of these simulations is the validation of our system by studying the quality of the classification.

Two scenarios are simulated :

- **Trajectory in a sparse environment:** The sparse environment is composed of a small number of objects. It consists only of buildings. It highlights the system's ability to detect buildings at different speeds.
- **The trajectory in a dense environment:** The dense environment is composed of a large number of objects. It consists of buildings, different models of parked vehicles and of moving vehicles. It represents a classical dynamic urban environment. The objective of the simulation is to assess the system's ability to detect a large number of different objects, moving or not.

Simulated vehicles have laser range finder sensors and are conducted by an operator. Simulations are running out hundreds of times in order to have a significant amount of data for a reliable statistical study.

1) sparse environment results: Figure 3 shows the classification of objects in time. Two classes are represented: building and another. One can observe that the system regularly detects a large number of objects in the environment ($t = 37s$ or $t = 360s$). These detections are due to disturbances corresponding to measurement errors (soil, sidewalks, etc.).

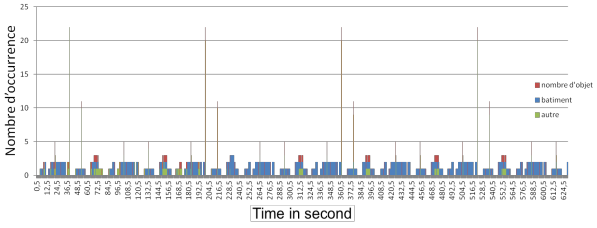


Figure 3. The evolution of the number of objects and their classification

These simulations show that in 12% of cases, a building is not detected by the system. 16% of the objects classified as buildings are false positives due to pitching of laser range finder during acceleration and braking.

2) dense environment results: Figure 4 shows the evolution of the number and type of objects detected during the simulation. We can see that increasing the number of objects does not cause a system overload. The list of identified objects is produced in less than 25 milliseconds. Note that the rate of not classified object's decreases compared to the simulation in a sparse environment. Indeed, because of the noise reduction ratio / useful information, the noise is less isolated and causes less disruption.

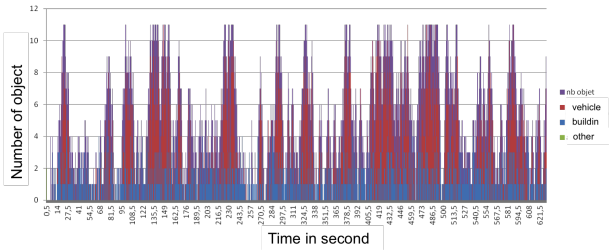


Figure 4. Distribution of classification

About misclassiffication, buildings are easier to detect than vehicles. In 82% of cases, a building is properly identified against 78% for vehicles. Similarly, the presence of the vehicle causes more false positives. This figure also shows that in these simulations, 12% of the vehicles were not detected by the system.

C. Experimentation results

Experimental results are based on the use of IRTESSET vehicles. We conducted several acquisitions campaigns to obtain sufficient data to study the behaviour of our approach in real situations. For these experiments, we make an acquisition campaign in Belfort city. The vehicle is driven by an operator throughout the circulation. The environment on the trajectory is composed of parked and moving vehicles, buildings and street furniture, such as sheltered bus stops.

The results discussed in this section correspond to the study of classifications made by our approach. Two classes of interest are defined : vehicles and buildings. The other detected objects are classified as "other" category and are considered as obstacles.

Figure 5 shows the evolution of the number of detected objects and their allocation over the time. We observe that the system limit is around 15 objects. Over this number, objects become too small in relation to the sensor resolution. This limit is due to the resolution proposed by the sensor. Finer resolution would detect more objects. We can also see that the system is able to quickly adapt to its environment. The number of detected objects can vary between appraisals. The largest identified variation is +/- 9 items in 0.25 seconds.

In terms of misclassification, buildings were detected in 78% of cases and that vehicles were detected in 72% of cases. The "other" category is the biggest generator of false positives. Among the non-detected objects, buildings and vehicles are below 11%. Street furniture that is often confused with the noise is difficult to identify. 72% of the objects belonging to this class are not identified.

 ISBN: 978-1-61208-391-9

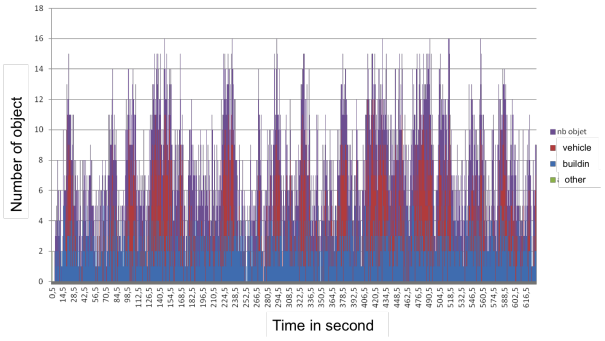


Figure 5. Classification of objects

We sought to quantify the duration of classification errors. These errors are either undetected objects, either false positives. Figure 6 shows that the error term is averaged higher for buildings than for vehicles. In almost 90% of cases, a vehicle is not detected between 0.25 and 0.5 seconds. Moreover, the distribution of misclassification of buildings is between 0.25 seconds and 1 second.

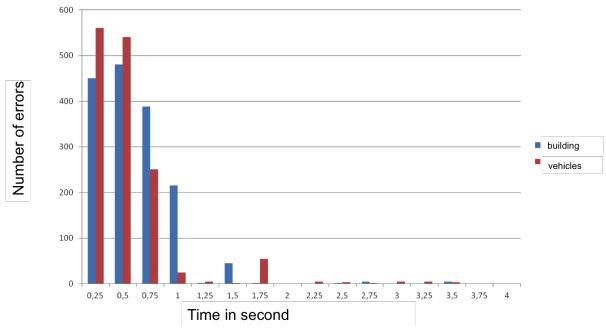


Figure 6. misclassification

We note that the real life results are worse than the simulation results, mainly due to the difference between the real and the simulated sensors. However, the simulation does not take into account the pure delays and the margins of error in the control actuators.

V. CONCLUSIONS

The paper presents a reactive agent approach for leader detection in platoon system through a generic and self-adaptive decision process. In this model, the environment is the central key element of agents system. It is the link between real world and the identification system. Agents population properties are observed to allow to choose and determine leader position in vehicle's perception and to separate it from other elements.

This solution has been successfully tested in simulation and in experimentation. The results obtained are encouraging to add and test the multi-sensor and plug and play ability of the system.

In order to continue this research, we are now working on a generic and self-adaptive approach for an agent based platoon control system.

ACKNOWLEDGMENT

This work was done with the support of the French ANR (National Research Agency) through the ANR-VTT *SafePlatoon* project (ANR-10-VPTT-011).

REFERENCES

[1] H. Huang, W. Jiang, W. Wang, Y. Cao, T. Zuo, and M. Li, "Simulation experiment of acquisition and tracking with beacon between the spatial movement platforms," Intelligent Control and Automation, 2008, pp. 7089–7093.

[2] R. Gibson, D. Hall, and J. Stover, "An autonomous fuzzy logic architecture for multisensor data fusion," Multisensor Fusion and Integration for Intelligent Systems, 1994, pp. 143–150.

[3] L. Guo, M. Zhang, Y. Wang, and G. Liu, "Environmental perception of mobile robot," Information Acquisition, 2006, pp. 348–352.

[4] B. Dafflon and F. Gechter, "Making decision with reactive multi-agent systems: A possible alternative to regular decision processes for platoon control issue," in Mexican International Conference on Artificial Intelligence, Nov. 2014.

[5] B. Dafflon, J. Contet, F. Gechter, and P. Gruer, "Toward a reactive agent based parking assistance system," The 24rd IEEE International Conference on Tools with Artificial Intelligence ICTAI, IEEE Computer Society, 2012.

[6] J.-P. Georgé, M. P. Gleises, F. J. Garijo, V. Noël, and J.-P. Arcangeli, "Self-adaptive coordination for robot teams accomplishing critical activities," PAAMS, 2010, pp. 145–150.

[7] G. DiMarzo-Serugendo, A. Karageorgos, O. Rana, and F. Zambonelli, "Engineering self-organising systems: Nature-inspired approaches to software engineering," Lecture notes in Atificial intelligence, vol. vol. 2977, 2004, p. 299.

[8] S. Shladover, "Ahs research at the california path program and future ahs research needs," Vehicular Electronics and Safety, 2008, pp. 4–5.

[9] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii, "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," Intelligent Transportation Systems, vol. 3, 2002, pp. 155–161.

[10] H. Fritz, "Longitudinal and lateral control of heavy duty trucks for automated vehicle following in mixed traffic: experimental results from the chauffeur project," Control Applications, vol. 2, 1999, pp. 1348–1352.

[11] R. Kunze, R. Ramakers, K. Henning, and S. Jeschke, "Organization and operation of electronically coupled truck platoons on german motorways," Springer Berlin / Heidelberg, 2009, pp. 135–146.

[12] B. Dafflon, F. Gechter, P. Gruer, and A. Koukam, "Vehicle platoon and obstacle avoidance: a reactive agent approach," IET Intelligent Transport Systems, 2013, pp. 257–264.

[13] M. El-Zaher, F. Gechter, P. Gruer, and M. Hajjar, "A new linear platoon model based on reactive multi-agent systems," The 23rd IEEE International Conference on Tools with Artificial Intelligence ICTAI, IEEE Computer Society, 2011.

[14] F. Gechter, J.-M. Contet, O. Lamotte, S. Galland, and A. Koukam, "Virtual intelligent vehicle urban simulator: Application to vehicle platoon evaluation," Simulation Modelling Practice and Theory (SIMPAT), 2012, pp. 103–114.

[15] B. Daffon, F. Gechter, J. Contet, A. Abbas-Turki, and P. Gruer, "Intelligent crossroads for vehicle platoons reconfiguration," International Conference on Adaptive and Intelligent Systems, 2011.

 ISBN: 978-1-61208-391-9

Interface Roles for Dynamic Adaptive Systems

Holger Klus

ROSEN Technology & Research Center GmbH
Am Seitenkanal 8
49811 Lingen (Ems), Germany
email: hklus@rosen-group.com

Dirk Herrling and Andreas Rausch

Technical University Clausthal
Julius-Albert-Straße 4,
38678 Clausthal-Zellerfeld, Germany
email: dirk.herrling@tu-clausthal.de
andreas.rausch@tu-clausthal.de

***Abstract*—Dynamic adaptive systems are systems that change their behavior according to the needs of the user at run time. Since it is not feasible to develop these systems from scratch every time, a component model enabling dynamic adaptive systems is called for. Moreover, an infrastructure is required that is capable of wiring dynamic adaptive systems from a set of components in order to provide a dynamic and adaptive behavior to the user. To ensure a wanted, emergent behavior of the overall system, the components need to be wired according to the rules an application architecture defines. In this paper, we present the Dynamic Adaptive System Infrastructure (DAiSI). It provides a component model and configuration mechanism for dynamic adaptive systems. To address the issue of application architecture conform system configuration, we introduce interface roles that allow the consideration of component behavior during the composition of an application.**

Keywords–dynamic adaptive systems; component model; adaptation; interface roles; application architecture awareness.

I. INTRODUCTION

Software-based systems are present at all times in our daily life. This ranges from our private life where nearly everyone owns and uses a smart mobile phone to large scale business applications and the public administration that is managed entirely by software systems. In every household, dozens of devices run software and a modern car will not even start its engine without the proper software. Some software systems have grown to be among the most complex systems ever made by mankind [1], due to their increase in size and functionality.

Through smaller mobile devices with accurate sensors and actuators and the ubiquitous availability of the Internet, the number of integrated devices in a large scale application has increased drastically within the last twenty years. These devices and the software running on them are used in organically grown, heterogeneous, and dynamic information technology (IT) environments. Users expect them not only to provide their primary services, but also to collaborate with each other and provide some kind of emergent behavior. The challenge is therefore to be able to build systems that are robust enough to withstand changes in their environment, deal with a steadily increasing complexity, and match requirements that might be defined in the future [2].

Due to the increasing complexity of large systems, be it in size or in functionality, those systems are no longer developed from scratch by one company. While the development usually takes place in a component-based way [3], it is usually split among a number of companies. Additional components for mobile devices are often developed against documented or reverse-engineered interfaces by independent developers.

To ease the development of dynamically integrateable components, a common component model is called for. The development of the DAiSI started in 2004 to address this issue [4]. Over the years a component model was defined that allows developers to implement a component for a dynamic adaptive system easily. In this DAiSI component model, every component contains an ordered set of component configurations which each map a set of required services to a set of provided services.

Additionally, a run-time infrastructure was described and implemented that can run and integrate DAiSI components by linking required services with compatible provided service and thus forming one ore more DAiSI applications. Compatibility has been only syntactical at first, requiring that for every method in the required service, a method with the same signature (name, parameters, return types, etc.) is defined [5]. The aspect was later extended to support semantic compatibility by additionally requiring equivalent behavior of each method [6].

Obviously, an application is more than just the sum of its components. This already becomes evident in very small examples. Consider cross country skiers and their trainer. A dynamic adaptive application connects vital data monitoring devices of the athletes to the management system of their trainer. In a competition with a competing team on the track, obviously not every athlete should be connected to every trainer. Also, the connection should not be made randomly. Each athlete should only be connected to the trainer that belongs to the same team. While it is possible to work around this issue by, e.g., ensuring in the implementation of the component that only athletes exchange data with trainers from the same team, this is just that – a work around.

An application architecture that is enforced by the infrastructure can define rules that can address the problem our athletes and trainers have. It can specify that only components of members of the same team are allowed to be bound to each other. More generically, the consideration of an application architecture during system configuration helps to ensure wanted, emergent behavior of dynamic adaptive systems. It does that by enabling application architects to limit the configuration space and thus prevent the connection of components that should not be connected. This paper will show a first step towards the introduction of an application architecture into the field of dynamic adaptive systems and how we integrate it with the DAiSI infrastructure.

 ISBN: 978-1-61208-391-9

The rest of this paper is structured as follows: In Section II, we will present an overview of other works in the field of dynamic adaptive systems. This is followed by an introduction to the DAiSI component model and the notation of DAiSI components in Section III. As a first step towards architecture conform configuration, we introduce interface roles in Section IV. The paper ends with a conclusion in Section V.

II. Related Work

Component-based development is one of the state-of-the-art techniques in modern software engineering. Components as units of deployment and their component frameworks provide a well-understood, solid approach for the development of large-scale systems. This is not surprising, considering that components can be added to, or removed from the system at design-time easily. This allows high flexibility and easy maintenance [3].

If components should be added to, or removed from a system at run-time things get a little bit more difficult, as techniques for this were not implemented in early component models. However, service oriented approaches allowed the dynamic integration of components at run-time. Those systems usually maintain a service directory and components entering the system register their provided, and query their required services at the directory. Once a suitable service provider is found for a required service, it can be easily connected to the component [7].

Service-oriented approaches are capable of handling dynamic behavior. Components that have not necessarily been previously known to the system can be integrated into the system. However, they have the uncomfortable characteristic that the system itself does not care for the dynamic adaptive behavior. The component needs to register and integrate itself. Also, it has to monitor itself if the used services are still available and adapt its behavior accordingly, if that is no longer the case. To address these issues a couple of frameworks have been developed to support dynamic adaptive reconfiguration.

CONIC was one of the first frameworks for dynamic adaptive, distributed applications. It provided a description technique that could be used to change the structure (and thus the architecture) of the integrated modules of an application. It was maintained through a central configuration manager [8]. With this description techniques, new component instances could be spawned and linked to each other.

Another framework, building on the knowledge gained through the CONIC development, was a framework for Reconfigurable and Extensible Parallel and Distributed Systems (REX). It provided support for dynamic reconfiguration in distributed, parallel systems. It visioned those systems as connected component instances with interfaces for which an own interface description language was defined. Components were considered as types, allowing multiple instances of any component to be present at run-time. The framework allowed the dynamic change of the number of running instances and their wiring [9], [10]. Both, the CONIC and the REX framework allowed the dynamic adaptation of distributed applications, but only through explicit reconfiguration programs for every possible occurring change.

This issue was addressed in [11]. They took a more abstract approach and defined valid application configurations. The system can then adapt itself from one valid application configuration to another, whenever the system changes. The declaration of reconfiguration steps became obsolete.

Another framework to build dynamic adaptive systems upon is ProAdapt. It is set in the field of service-oriented architectures and reacts to four classes of situations:

- Problems that stop the execution of the application
- Problems that require the execution of a non-optimal system configuration
- Arising of new requirements
- Providing of services with a better service quality

ProAdapt is capable of replacing certain services and can, together with its service composition capabilities, replace composed services [12].

In [13], [14], a framework for the dynamic reconfiguration of mobile applications on the basis of the .NET framework was introduced. Applications are composed of components, and application configurations are specified initially in XML. A centralized configuration manager interprets this specification and instantiates and connects the involved components. The specification can include numerous different configurations which are distinguished through conditions under which they apply. The framework monitors its surroundings with the help of a special *Observer* component and evaluates which application configuration is applicable. The framework allows the dynamic addition and removal of components and connections.

In [15] the authors presented a solution to ensure syntactical and semantical compatibility of web services. They used the Web Service Definition Language (WSDL) and enriched it with the Web Service Semantic Profile (WSSP) for the semantical information. Additionally they allowed an application architect to further reduce the configuration space through the specification of constraints. While their approach is able to solve the sketched problem of preventing the wiring of components that should not be connected, they only focus on the service definition and compatibility. Our DAiSI approach defines an infrastructure in which components are executed that implement a specific component model. We do want to compose an application out of components that can adapt their behavior at run-time. We achieve this by mapping sets of required services to sets of provided services and thus specifying which provided services depend on which required services. The solution presented in [15] does not offer a component model. All rules regarding the relation between required and provided services would have to be specified as external constraints. The authors in [16] provided a different solution to ensure semantic compatibility of web services. However, the same arguments as for [15] regarding the absence of a high level component model hold true.

III. The DAiSI Component Model

This section will introduce the foundations of the DAiSI component model. As already briefly mentioned in the introduction of this paper, DAiSI components communicate with each other through services. Different component configurations map which required services are needed by the associated provided services. Figure 1 shows a sketch of a DAiSI component with some explanatory comments for an athlete in the biathlon sports domain.

 ISBN: 978-1-61208-391-9

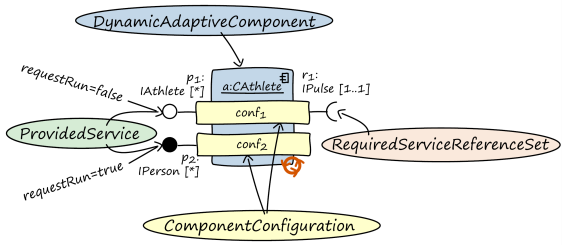


Figure 1. Example notation of a DAiSI component with explanatory comments.

A component is depicted as a rectangle, in this example of a light blue color. Component configurations are bars that extend over the borders of the component and are depicted in yellow here. Associated to the component configurations are the provided and required services. The notation is similar to the Unified Markup Language (UML) lollipop notation [17] with full circles resembling provided, and semi circles representing required services. A filled circle indicates that the associated service is directly requested by the end user and thus should be provided, even if no other service requires its use.

Figure 1 shows the *CAthlete* component, consisting of two component configurations: conf_1 and conf_2. The first component configuration requires exactly one service variable r_1 of the *IPulse* interface. The second component configuration does not require any services to be able to provide its service p_2 of *IPerson*. The service could be used by any number of service users (the cardinality is specified as $*$). The other component configuration (conf_1) could provide the service p_1 of the type *IAthlete*, which could again be used by any number of users.

Figure 2 shows the DAiSI component model as an UML class diagram [17]. The component itself, represented as the light blue box in the notation example, is represented by the *DynamicAdaptiveComponent* class. It has three types of associations to the *ComponentConfiguration* class, namely *current*, *activatable*, and *contains*. The *contains* association resembles the non-empty set of all component configurations. It is ordered by quality from best to worst, with the best component configuration being the most desirable, e.g., because of best service qualities of the provided services. The order is defined by the component developer. A subset of the contained are the *activatable* component configurations. These have their required services resolved and could be activated. An *active* component configuration produces its provided services. At run-time, only one or zero component configurations per component can be active. The active component configuration is represented by the *current* association in the component model, with the cardinality allowing one or zero current component configurations for each component.

The required services (represented by a semi circle in the component notation in Figure 1) are represented by the *RequiredServiceReferenceSet* class. Every component configuration can declare any number of required services. Those that are resolved are represented by the *resolved* association. The cardinalities of the required service are stored in the attributes *minNoOfRequiredRefs* and *maxNoOfRequiredRefs*. Provided services (noted as full circles on the left hand side in

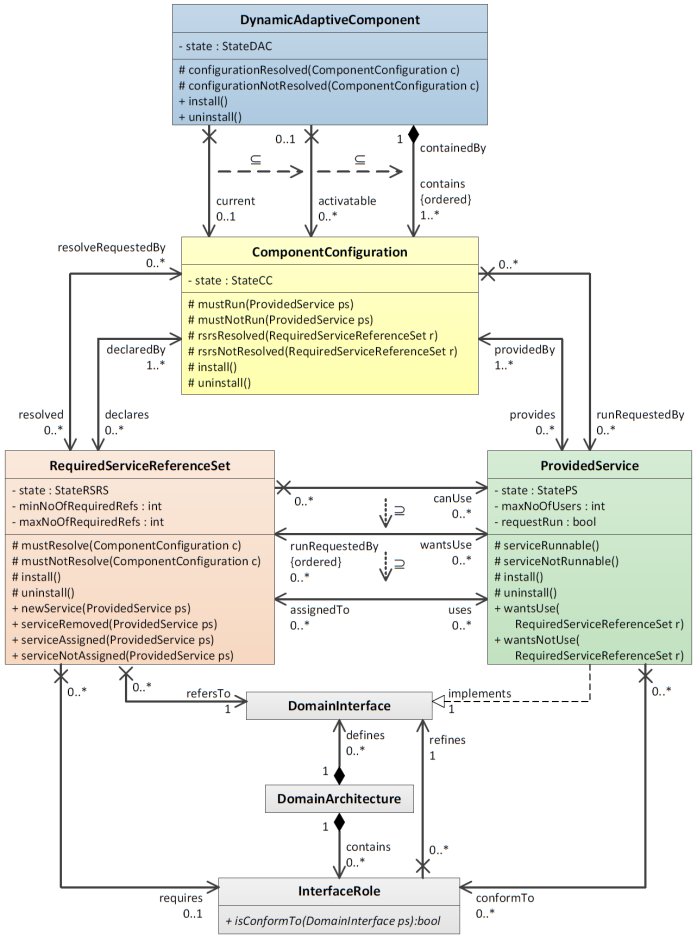


Figure 2. DAiSI component model.

Figure 1) are represented by the *ProvidedService* class. They can be associated to more than one component configuration, if more than one component configuration provides the same service. The *runRequestedBy* association is relevant at run-time and resembles the component configuration that actually wants the provided service to be produced.

Not all provided services can be used any number of times. The attribute *maxNoUsers* indicates the maximum number of allowed users. The flag *requestRun*, represented by the full circle being filled with black in the component notation, indicates that the service should be produced, even if no other service requires its use. This is typically the case for services that provide graphical user interfaces or that provide some functionality directly requested by the end user.

The provided and required service, more precisely their respective classes in the component model, are associated with each other through three associations. The first association *canUse* represents the compatibility between two services. If a provided service can be bound to the service requirement of another class, these two are associated through a *canUse* association. A subset of the *canUse* association is *wantsUse*. At run-time, it resembles a kind of reservation of a particular provided service by a required service reference set. It does not already use the provided service, but would like to use it. After the connection is established and the provided service

satisfies the requirement, they are part of the *uses* association which represents the actual connections. All classes covered to this point implement a state machine to maintain the state of the DAiSI component. If you want to know more about the state machines and the configuration mechanism, please refer to our last years paper [18].

To this point, we have covered the building blocks of a DAiSI component. An application in a dynamic adaptive environment is composed of any number of such components that are linked with each other through services. Those services are defined through *DomainInterfaces*. Required services (represented by the *RequiredServiceReferenceSet* class) refer to exactly one domain interface, while provided services (represented by the *ProvidedService* class) implement a domain interface. The set of all defined domain interfaces composes the *DomainArchitecture*. The interface roles, which will be presented in the next section, are contained by the domain architecture. They refine domain interfaces and are required by any number of required service reference sets. Any provided service can conform to an interface role. However, this is not a static information, but changes during run-time. The next section will explain why.

IV. Interface Roles

With the *RequiredServiceReferenceSet* class many component local requirements can be specified. However, this is not sufficient for self-organizing systems.To illustrate the problem, let us consider Figure 3. It shows a DAiSI component for an athlete in the biathlon sports domain. It does specify one component configuration which provides a service of the type *IAthlete* and requires two *IStick* services to be able to do so. The provided service calculates the current skiing technique and needs measurement data of the sticks movements, which is provided by the two required services. However, with the component model as presented in the previous paragraphs a binding between only the left ski stick with both required service reference sets would be possible and allow the component to run. Of course the domain interface *IStick* does provide a method to query at which side a ski stick is being used. However, this information is not considered in the configuration process. Obviously, the *IAthlete* service can not perform as expected as the measurement data of the right ski stick are missing.

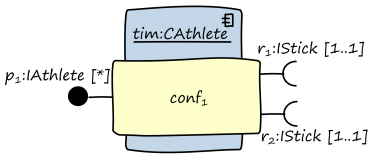


Figure 3. A DAiSI component for a biathlon athlete.

There are numerous other examples in which return values of domain services have to be considered in order to establish the desired system configuration. For that reason, we extended the component model by the class *InterfaceRole*. In our previous understanding, provided and required services were compatible, if they referred to the same domain interface. Those interfaces can be seen as a contract between service provider and service user. We now extended this contract by interface roles. An interface role references exactly one domain interface and can define additional requirements regarding the return values of specific methods defined in that domain interface. A provided service only fulfills an interface role if it implements the domain interface and as well complies to the conditions defined in the interface role. Consequently, a required service reference set not only requires compatibility of the domain interface, but also of the interface role to be able to use a provided service.

Figure 4 shows the same DAiSI component as Figure 3, but with specified interface roles. With this addition it can be ensured that the athlete component in fact is connected with one left and one right stick. The *LeftStickRole* interface role refines the *IStick* domain interface and compares the return value of the method that returns the side of the ski stick is used on against a reference value for left ski sticks. This could be implemented by a method called *getSide():String* and the return value would be compared against the string “left”. The interface role *RightStickRole* can be implemented accordingly.

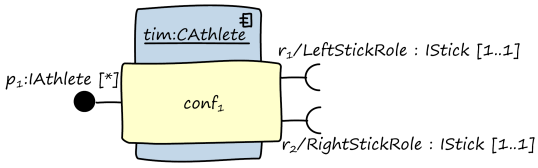


Figure 4. A DAiSI component for a biathlon athlete with interface roles.

This solution introduces new challenges for the configuration process of dynamic adaptive systems. Was it previously sufficient to connect a pair of required service reference set and provided service, this decision has to be monitored now. As the interface roles take return values of services into account, the fulfillment of an interface role is not static. The provided service supposedly conforming to the interface role has to be evaluated either cyclically, or event based whenever relevant return values change. For our implementation, we took a cyclic approach, however in [6] we described a way to re-evaluate the semantic compatibility of services whenever return values change equivalence classes.

V. Conclusion

This paper presented an extended version of the DAiSI framework. The key aspect that the developer does not need to implement the adaption behavior himself, has been prevailed. While the system configuration, more precisely the component wiring, in older versions of DAiSI and other dynamic adaptive systems was only considering syntactic and semantic compatibility, the newest findings enable the developer to specify interface roles. These open the possibility to consider return values of services during system configuration, which was not possible before. We implemented the framework in Java and a slightly limited version in C++. Components of both framework implementations can be linked together, because of an underlying CORBA layer.

Interface roles are obviously just the first step towards application architecture conform system configuration. In the near future, we will extend the approach to support an architecture description that allows the specification of constraints that are not component local, as the interface roles are.

 ISBN: 978-1-61208-391-9

REFERENCES

[1] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau, "Ultra-Large-Scale Systems - The Software Challenge of the Future," Software Engineering Institute, Carnegie Mellon, Tech. Rep., June 2006. [Online]. Available: http://www.sei.cmu.edu/uls/downloads.html

[2] J. Kramer and J. Magee, "A rigorous architectural approach to adaptive software engineering," Journal of Computer Science and Technology, vol. 24, no. 2, 2009, pp. 183–188.

[3] C. Szyperski, Component Software: Beyond Object-Oriented Programming, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] D. Niebuhr, C. Peper, and A. Rausch, "Towards a development approach for dynamic-integrative systems," in Proceedings of the Workshop for Building Software for Pervasive Computing, 19th Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA), Nov 2004. [Online]. Available: http://sse-world.de/index.php/download_file/view_inline/157/

[5] H. Klus, D. Niebuhr, and A. Rausch, "A component model for dynamic adaptive systems," in Proceedings of the International Workshop on Engineering of software services for pervasive environments (ESSPE 2007), A. L. Wolf, Ed. Dubrovnik, Croatia: ACM, sep 2007, pp. 21–28, electronic Proceedings. [Online]. Available: http://sse-world.de/index.php/download_file/view_inline/79/

[6] D. Niebuhr, Dependable Dynamic Adaptive Systems. Verlag Dr. Hut, 2010. [Online]. Available: http://www.dr.hut-verlag.de/9783868536706.html

[7] M. P. Papazoglou, "Service-oriented computing: Concepts, characteristics and directions," in Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on. IEEE, 2003, pp. 3–12.

[8] J. Magee, J. Kramer, and M. Sloman, "Constructing distributed systems in conic," Software Engineering, IEEE Transactions on, vol. 15, no. 6, 1989, pp. 663–675.

[9] J. Kramer, "Configuration programming-a framework for the development of distributable systems," in CompEuro'90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering. IEEE, 1990, pp. 374–384.

[10] J. Kramer, J. Magee, M. Sloman, and N. Dulay, "Configuring object-based distributed programs in rex," Software Engineering Journal, vol. 7, no. 2, 1992, pp. 139–149.

[11] I. Warren and I. Sommerville, "Dynamic configuration abstraction," in Software EngineeringESEC'95. Springer, 1995, pp. 173–190.

[12] R. R. Aschoff and A. Zisman, "Proactive adaptation of service composition," in Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012 ICSE Workshop on. IEEE, 2012, pp. 1–10.

[13] A. Rasche and A. Polze, "Configurable services for mobile users," in Object-Oriented Real-Time Dependable Systems, 2002.(WORDS 2002). Proceedings of the Seventh International Workshop on. IEEE, 2002, pp. 163–170.

[14] ——, "Configuration and dynamic reconfiguration of component-based applications with microsoft. net," in Object-Oriented Real-Time Distributed Computing, 2003. Sixth IEEE International Symposium on. IEEE, 2003, pp. 164–171.

[15] T. Kawamura, J.-A. De Blasio, T. Hasegawa, M. Paolucci, and K. Sycara, "Public deployment of semantic service matchmaker with uddi business registry," in The Semantic Web ISWC 2004, ser. Lecture Notes in Computer Science, S. McIlraith, D. Plexousakis, and F. van Harmelen, Eds. Springer Berlin Heidelberg, 2004, vol. 3298, pp. 752–766. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-30475-3_52

[16] T. Haselwanter, P. Kotinurmi, M. Moran, T. Vitvar, and M. Zaremba, "Wsmx: A semantic service oriented middleware for," in B2B Integration, International Conference on Service-Oriented Computing. Springer, 2006, pp. 4–7.

[17] OMG, OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.4.1, Object Management Group Std., Rev. 2.4.1, August 2011. [Online]. Available: http://www.omg.org/spec/UML/2.4.1

[18] H. Klus and A. Rausch, "Daisi - a component model and decentralized configuration mechanism for dynamic adaptive systems," in ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications, 2014, pp. 27–36.

 ISBN: 978-1-61208-391-9

Ecosystems Enabling Adaptive Composition of Intelligent Services

César A. Marín

Service Systems Research Group, MBS.
The University of Manchester.
Manchester M15 6PB, United Kingdom
Email: cesar.marin@manchester.ac.uk

José Barbosa and Paulo Leitão

Polytechnic Institute of Bragança.
5301-857 Bragança,
Portugal
Email: {jbarbosa,pleitao}@ipb.pt

***Abstract*—Intelligent services are roughly defined as pieces of software with the capabilities of problem-solving and autonomous composition of solutions, e.g., composition of manufacturing processes. They are heterogeneous and distributed by design, however in industrial settings they are constrained to local interactions with limited room for adaptation due to the need to lower the interoperability barrier. In this paper, we present an approach where collections of intelligent services are treated as ecosystems, using food chains, environments and migration to enable adaptive compositions that generate solutions with a higher service value chain. We present a set of experiments demonstrating how a distributed ecosystem achieves compositions of solutions with higher service value chains while balancing the load and diversity of intelligent services across the ecosystem via self-organisation. This supports the claim that implementations of intelligent service based systems (ISBS) as ecosystems could bring substantial benefits to industrial applications.**

Keywords–Ecosystem; Intelligent services; Adaptive composition.

I. INTRODUCTION

Recent developments in industrial applications of agent [1] and service [2] technologies have seen the combination and redefinition of the term *intelligent services*. Intelligent services have been defined as ”independent pieces of software that are expected to provide a particular result, either produced by the intelligent service itself or by requesting support from other intelligent services” [3]. They are used to compose solutions to problems in an autonomous manner, e.g., composition of manufacturing processes [3] [4]. By definition, intelligent services accommodate the existence of heterogeneous implementations such as software agents, web services, or any of their combinations. This creates an interoperability barrier that is lowered by the utilisation of a central Enterprise Service Bus (ESB) for multi-protocol communication [5].

The result is an ISBS distributed by design, but centralised by implementation due to the ESB because all intelligent services have to connect to it for communication. Consequently, the intelligent services are constrained to local interactions with limited room for adaptation. This calls for an approach that unlocks the potential of intelligent services by reaching outside their centralised implementation while benefiting from an environment where the interoperability barrier has been lowered.

The contribution of this paper is an approach where ISBS are treated as ecosystems, where food chains, environments and migration trigger an adaptive composition that obtains solutions that benefit from higher a service value chains across ecosystems. Our experiments demonstrate that the service value chain of composed solutions increases while the system regulates its diversity. This sustains the claim that industrial applications of intelligent services could benefit from using our ecosystem approach to achieve more stable, balanced, efficient and adaptive distributed systems for their own support.

The remaining of this paper is structured as follows: Section II provides more details of ISBS. Section III introduces the approach of using ecosystems as a way to model intelligent service compositions. Section IV explains the experiments carried out and analyses the achieved results. Section V discuss related work done in the area. Finally, Section VI round up the paper with a conclusion.

II. INTELLIGENT SERVICE BASED SYSTEMS

Cyber-Physical Systems are seen as the way to support the fourth industrial revolution where all devices that act on the industry will be connected. These systems promote the decentralisation of the control over a set of distributed entities where, through their cooperation, the global system behaviour is achieved. One possibility to develop the interface layer of these systems is by the adoption of functionality exposure to others using service oriented architecture. To this extend, the current use of services is not sufficient, since in general they do not exhibit intelligence which could allow them to make on-the-fly adaptations as the surrounding constraints change. In this way a different view of services is needed where behaviour is embedded.

We take the definition of *intelligent services* from [3] which defines them as ”independent pieces of software that are expected to provide a particular result, either produced by the intelligent service itself or by requesting support from other intelligent services”. This implies that intelligent services possess the capabilities of problem-solving and autonomous composition of solutions, e.g., process composition in manufacturing. Particularly, intelligent services are not bound by any specific technology nor platform, but rather can be seen as autonomous entities that expose their internal functionalities as services. In practice, they are the combination of two worlds: agents providing autonomy and intelligence, and services offering ease of aggregation [6].

Autonomous agents can be used to provide the services with the ”intelligent” part allowing the reasoning and adaptation behind the service encapsulation. To this extent, basically three types of combinations can be envisioned [7]. The first

 ISBN: 978-1-61208-391-9

approach is to use gateways for semantic translation from the agent world to the service world; Another approach is to encapsulate single agents as services, thus having a direct access to other services; The last approach is to use service-oriented agents that not only share services as their major form of communication, but also complement their own goals with external provided services.

By definition, intelligent services do not rely on any specific agent-service combination, but rather allow heterogeneous implementation. Therefore, they must rely on a common information exchange platform to lower any interoperability barrier [5]. This can be achieved by means of an ESB, which provides the basic functionalities for communications exchange such as routing and conflict resolution [8], see Figure 1. Transposing this topology into a real world situation, cases are found where a company is composed by multi-site delegations or factories, each one having its own ESB, and thus having the need to share information and services being offered in other companies' ESBs. Different cases can also be found in situations where a set of heterogeneous companies create a working cluster where information and services need also to be shared.

One of the issues found in implementations of intelligent services is the lack of mechanisms that enable the dynamic replacement of intelligent services in order to better accommodate the structural adaptation around the ESB. This urges the need to create a network of cooperative ISBS in which the offered services of one ISBS migrates into another one according to where they are most needed. This structural change at the ESB level can be achieved by drawing a parallel to natural ecosystems, thus enabling the creation of an ecosystem of ISBS where services migrate from one ISBS to another in their search for better compositions.

III. Ecosystems for Intelligent Services

Natural ecosystems are typical examples of adaptive and self-organising systems where local interactions enable the emergence of complex behaviours [9]. Current implementations of ISBS, e.g., [4], exhibit local interactions which limit their capability for adaptation, a restriction that can be exploited by drawing the parallel between natural ecosystems and ISBS. This enables adaptation of intelligent service compositions across a set of instances of ISBS. Our approach considers three elements of a natural ecosystem that in previous studies have proven successful [9]: *food chains*, *environment* and *migration*. These are explained in the following subsections.

A. Intelligent Service Composition as Food Chains

A food chain is the collection of species in which energy and resources flow from one species to another [10]. Yet species may not only consume from one single species, they may belong to overlapping food chains. Thus the participation of a species in a food chain varies according to the species participation in another food chain, creating shifting networks of energy and resource flows [11]. Consequently, food chains are dynamic in such a way that they are created, changed, replaced, dissolved or re-created continually over time. In terms of intelligent services, a food chain is the equivalent of a full composition of intelligent services where a service value chain is generated by aggregating the individual service values from the first intelligent service in the chain up to the last one.

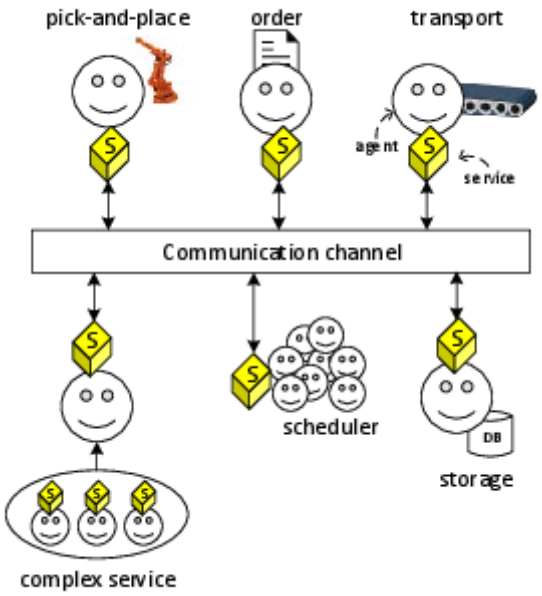


Figure 1. Intelligent services connected to an ESB.

We see each individual in an ecosystem as a member of a specific species which produces a resource of one type and consumes a resource of another type. In terms of ISBS, an individual is the intelligent service itself and the resource is the functionality being offered and consumed by intelligent services. Now let us define the basics for intelligent service composition under the ecosystem approach. For the sake of clarity of and focus on the ecosystem approach, we do not consider any semantic matching or similar. We simply assume that such approaches can be incorporated, e.g., [12].

A link in a composition is formed when the produced resource of an intelligent service is consumed by another intelligent service in such a way that both intelligent services perceive a benefit in maintaining this producer-consumer relationship over others. This is represented as

$$l = i \odot j \tag{1}$$

where i and j are intelligent services and j consumes a resource i produces. The symbol $\odot$ is used to denote that the link formed by two intelligent services represents a resource flow from a producer to a consumer. The operator $\odot$ is not commutative.

The value of a link between the intelligent services i and j is defined as a function in the following way:

$$\nu(l) = \nu(i) + \nu(j) \tag{2}$$

where ν is a function that returns the value of the functionality offered by an intelligent service or a composition of them. For the purpose of this paper such a value is a number used to compare and calculate the value of a composition, the higher the value the better.

A path is a succession of links starting from an intelligent service i to a final intelligent service j where the path is the minimum set of links required to connect them; thus

$$\pi^n(j) = l_0 \odot l_1 \odot \ldots \odot l_{n-1} \tag{3}$$

 ISBN: 978-1-61208-391-9

where $n > 0$ and l_0 begins with intelligent service i and l_{n-1} ends with intelligent service j; that is, $l_0 = i \odot \ldots$ and $l_{n-1} = \ldots \odot j$. It is possible to write $i\,\pi^n\,j$ to indicate that there exists a path from i to j consisting of n links. Notice that $i\,\pi^1\,j \equiv l_0 \equiv i \odot j$.

The notion of *predecessor* and *successor* is used to determine a relative position in a path. Let i and j be intelligent services, it is said that "i is a predecessor of j" or "j is a successor of i" if $i\,\pi^n\,j$ is a valid path.

The value of a path is calculated by recursively adding up the value of the constituting links from the intelligent service i to j in the following way:

$$\nu(\pi^n(j)) = \nu\left(\pi^{n-1}(k)\right) + \nu(j) \tag{4}$$

where $n > 0$ and k is the immediate predecessor of j in the path.

A composition of intelligent services, where the final solution is given by intelligent service j, is the collection of paths connecting intelligent services to the final provider j. This is expressed in the following way:

$$\Pi^n(j) = \{i\,\pi^n\,j\,,\,\forall\,i\} \tag{5}$$

where $n > 0$. It possible to write $i\,\Pi^n\,j$ to denote a composition where i is at the beginning of a constituting path of the composition and j is the final intelligent service providing the final solution. In the simplest case, $\Pi^1(j) \equiv \pi^1(j) \equiv i \odot j$.

Finally, *the service value chain* of the composition $\Pi^n(j)$ is then calculated by adding the value of all the paths where j is the common successor. This is represented in the following way:

$$\nu(\Pi^n(j)) = \sum^{\Pi^n(j)} \nu(\pi^n(j)) \tag{6}$$

where $n > 0$. The service value chain is created as resource flow from a single intelligent service all the way through to the final producer of a composite service. The service value chain is used within ISBS to enable the evaluation of compositions, which in turn allow intelligent services to make decisions about who they interact with in order to increase the value of their produced resource, i.e., their offered functionality.

B. Distributed Environment of Migrating Intelligent Services

The environment is an essential element in a natural ecosystem. Without it, species would struggle to survive since it enables them to search for resources to consume. In the context of ISBS, the ESB is the equivalent to the environment. Without it, intelligent services would struggle to interact due to their heterogeneous implementations (see Section II). In natural ecosystems, migration allows species to effectively switch from one environment to another according to the abundance of resources they need. In the context of the ISBS, migrating from one ESB to another would allow intelligent services to relocate to a different environment according to the abundance of resources provided by the intelligent services there.

Our approach considers the environment, cf. [9], as a virtual surface where intelligent services behave as individuals, wandering across and encountering others in order to interact; it mediates interactions and allows intelligent services to forage for resources of their interest. Nevertheless, they keep on exploring the environment for better resources while returning to areas where they have had favourable interactions in the past. The result of this is a dynamic setting where all intelligent services are in motion interacting with others crossing their path.

We see a distributed environment as a collection of interconnected environments forming a network where each of the environments has its own set of intelligent services inhabiting it. Intelligent services then can move from one environment to another in a similar way to species in a natural ecosystem migrate from one environment to another. Since the intelligent services' interactions are local to the environment they occupy, the migration is managed by the environments (i.e., the ESB). Therefore, each environment considers two criteria for enabling migration: 1) the selection of the intelligent service to migrate, and 2) the destination environment.

1) Selection of intelligent service to migrate. We consider two conditions for migration for each type of intelligent service: a) *resource value*, migrate the intelligent service that individually contributes less to any service value chain, and b) *past interactions*, migrate the intelligent service that has interacted less in a certain period of time.

The first condition focuses on moving the intelligent service that, because of its low service value, is likely to be disfavoured as preferred producer, as it is probable for not being selected to be interacted with. By making it migrate, the remaining intelligent services do not waste time in interactions that will not be profitable thus increasing the overall chance of having a greater service value chain.

The second condition focuses on past interactions. We consider each intelligent service possessing a rolling memory with which they only remember the last n interactions. Therefore, if an interaction with a preferred producer does not repeat before fading away, the preferred producer will be forgotten. This characteristic motivates intelligent services to keep on interacting with their peers as a way to remember preferred interactions while keeping exploring for new and possibly better resources. Consequently, the second condition for migration focuses on that intelligent service that has interacted less in a certain period of time since its exploration has resulted less fruitful. With migration, such intelligent service is given the opportunity to explore in a different environment and thus increasing the chance of being preferred somewhere else.

In both migration conditions the main goal is to allow intelligent services to move to an environment where they can be more useful. Both migration conditions alter the composition of intelligent services and thus the service value chain across the distributed environment. Therefore, the *stability of the compositions* and the *service value chains* are analysed in Section IV.

2) Selection of destination environment for migration. The main concern here is to balance the overload between the environments that constitute the ecosystem, which is based on the current load of intelligent services per produced resource type. The environments are able to calculate their current load and share it with each other, comparing the frequencies of intelligent services local interactions between pairs of environments. The environment pair with the highest difference between frequencies will trigger the migration of an intelligent service between each other, migrating from the high loaded

 ISBN: 978-1-61208-391-9

environment to the less loaded.

```
1: threshold, migrationCondition
2: for all IntelServiceTypes do
3:    maxInteraction ← max(IntelServType)
4:    minInteraction ← min(IntelServType)
5:    if maxInteraction − minInteraction > threshold then
6:       if migrationCondition == resourceValue then
7:          IntelServToMigrate ← min(lowestResourceValue)
8:       end if
9:       if migrationCondition == pastInteractions then
10:         IntelServToMigrate ← min(lowestInteraction)
11:      end if
12:      migrate(IntelServToMigrate)
13:   end if
14: end for
```

Figure 2. Migration decision mechanism.

As a consequence of migration, new service compositions emerge in each environment due to the presence of new intelligent services. Likewise, existing compositions are forced to adapt to the new circumstances. Furthermore, the diversity of intelligent services in each environment is affected. In biology, it is well known that the diversity of an ecosystem is a principal ingredient for adaptation and resilience. Analyses of diversity, from the biology point of view, have been carried out in other works to estimate the resilience of information systems [13] [14]. Therefore, the *diversity* of the environments will be measured to estimate the evenness of intelligent services across the distributed environment. See Section IV for more details.

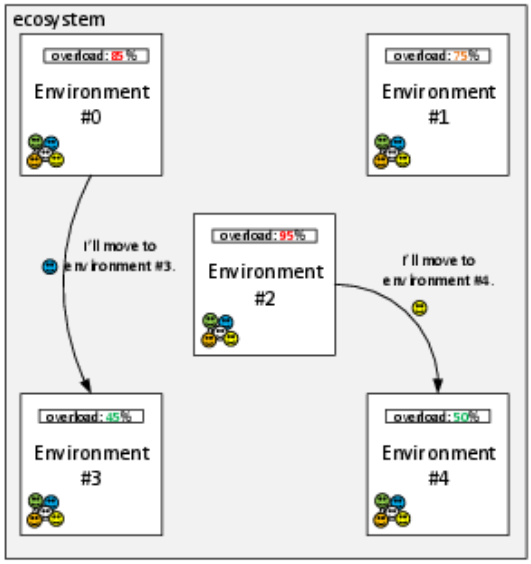


Figure 3. Ecosystem for intelligent services.

3) Combination of migration criteria. The algorithm of the described criteria is shown in Figure 2. It guarantees that at least one intelligent service producing each resource type remains in each environment. This is done to allow at least one full composition to emerge in each environment. Finally, Figure 3 depicts how the distributed ecosystem would look like using five environments. As an example, the interaction load annotated in the figure is used to indicate the environment destination for migration. In the example, it is possible to observe that environment #0 and #2 have high load levels as opposing to environments #3 and #4. Therefore based on the migration criteria, an intelligent service can migrate from a highly loaded environment to one less loaded, trying to balance the environment load while offering the intelligent services the opportunity to maximise their resource value. This triggers both the emergence of new compositions and the adaptation of existing ones.

By having an intelligent service selection criteria and a migration environment selection, the overall ecosystem can remain stable, avoiding disparity of overloaded and underused environments. Despite this environment overload distribution, the generated service value chain can also experience an increase, since intelligent services now have mechanisms to keep interacting with preferred services while constantly seeking for new opportunities to evolve and adapt. In such way, the previously given definition for intelligent services can now be extended by stating that those now have capabilities to switch between "working" environments aiming to increase their usefulness.

IV. Experiments and Analysis of Results

In order to analyse the advantage of introducing the ecosystem elements of 1) food chains, 2) environment and 3) migration, we use a set of metrics to measure 1) the variation of the service value chains, 2) the stability of compositions, and 3) the diversity of intelligent services across the distributed environment. For this purpose we use the following metrics:

Service value chain. It shows the statistical distribution of the service value chain collected by all the intelligent services acting as top consumers of the composition. The higher and more stable the better.

Stability. It calculates the accumulated number of times the service value chain changes from one simulation step to the next one. The lower and more stable the better.

Diversity. It uses the normalised Shannon Index to measure the level of diversity at a given time in one environment as in [13]. The median of all the normalised Shannon Indexes is then calculated to estimate the diversity across the distributed environment. This is expressed as follows:

$$H' = \frac{-\sum_{i=1}^{S} p_i \ ln(p_i)}{ln(S)} \tag{7}$$

where S is the total number of species (i.e., intelligent service types), p_i is the proportion of individuals of species i (i.e., intelligent services of type i) in the population. The higher the value, the more evenly the species are represented in the environment. The lower the value, the less even is the representation. A normalised diversity of 1 indicates all species have exactly the same number of representatives in that environment. A normalised diversity of 0 indicates all intelligent services in that environment belong to the same type. Therefore, the higher and more stable the value is the better is considered.

A. Experimental Setup

For the experimental test-bed described in this work, Netlogo platform was selected since it aggregates in an overall

 ISBN: 978-1-61208-391-9

manner a good combination of GUIs, ease of programming and an extensive documentation support [15]. We ran two experiments, one per migration condition. **Experiment 1: resource value. Experiment 2: past interactions**. For each experiment, we ran 30 simulations which then were used to calculate the median across the same simulation step. This way, we can appreciate the central behavioural pattern with statistical significance. The median is used as the central representation point because is does not assume any distribution in the sample data.

Each simulation consists of the following consecutive stages:

- Running five independent ecosystems for 1,000 simulation steps, for letting the ecosystems converge to a service value chain. This number of simulation steps was empirically determined to be more than sufficient to detect any convergence in the individual environments.
- Enabling migration between the five environments, rendering a distributed environment (cf. a fully connected network), for 2,000 simulation steps. The number of simulation steps for this stage was empirically determined as well in order to allow the simulations to run for a longer period and be able to notice any trend.

B. Migration Conditions Increase the Service Value Chain

Figures 4 and 5 show the service value chain value of experiment 1 and 2 respectively. It can be immediate noticed in both cases that the convergence to a (local) optimum is reached during the first stage. Even when there is no connection between environments, these are in continual internal dynamism because intelligent services interact with whoever they encounter in their environment, keep looking for preferred producers, forget useful interactions, and interact again. Also, notice that the service value chain stabilises even under such dynamism because negative changes are absorbed by these interactions, cf. [4].

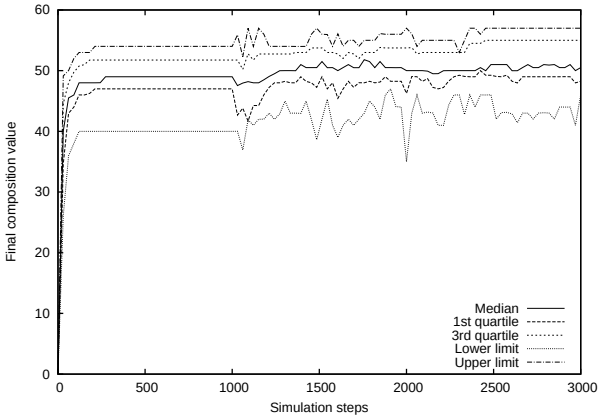


Figure 4. Service value chain value with resource value as migration condition.

The moment migration is enabled, the service value chain immediately varies and the median tends to take slightly higher values. However, when using the resource value as the migration condition, the distribution of the service value chain (Figure 4) tends to be slightly more compact than when

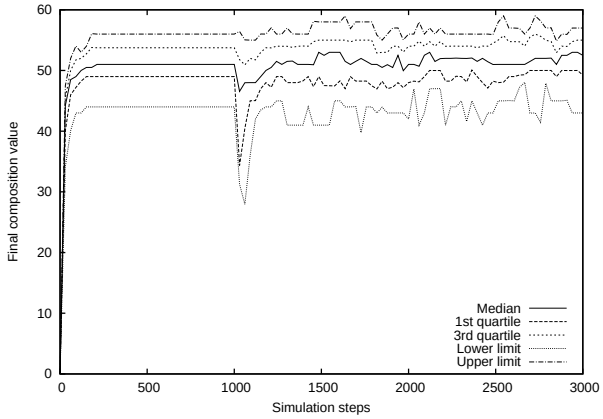


Figure 5. Service value chain with past interactions as migration condition.

past interactions is used (Figure 5). Additionally, migration with past interactions (Figure 5) shows a sharp drop in the service value chain right after migration is enabled. This occurs because in this mode migration does not care about resources, whereas in the other case, intelligent services still try to optimise the resource they produce. Nonetheless, the median shows a slight improvement in the service value chain when migration is enabled.

C. Past Interactions Render the Ecosystem More Stable

Figures 6 and 7 depict the stability of the service value chain of experiment 1 and 2, respectively. In both cases, the stability during the first 1000 simulation steps stays under 15 number of changes, this means that environment tend to find and keep an optimum service value chain. However, the moment migration is enabled using resource value, as shown in Figure 6, the number of changes tends to increase and its distribution to widen over time. This is because the motivation of the intelligent services to move is their resource value only, thus making composition more volatile.

In the case of migration with the past interactions condition, as depicted in Figure 7, the number of changes tends to increase as well, but its distribution widens considerably less than its counterpart. This is because the intelligent services that migrate are the intelligent services that interact less in an environment. Consequently, their own resource value is not that essential towards the final composition in that environment. Therefore, if they migrate the remaining intelligent services hardly notice the change.

D. Past Interactions Maintain Ecosystem Diversity

Figures 8 and 9 present the diversity across the environments. During the first stage of simulation, both cases show a normalised diversity of 1, meaning that all intelligent service types have exactly the same number of individuals in each environment. This value is expected since no migration has been enabled yet. The moment migration is enabled, a drop in diversity immediately occurs in both cases. This is because initially there is no order regarding where the intelligent services are better required. As the simulations progress and the system starts to adapt, the levels of diversity increase because the environments acquire patterns in terms of proportions of intelligent services per type.

However, when resource value is used as the migration condition (Figure 8), the distribution of diversity widens over

 ISBN: 978-1-61208-391-9

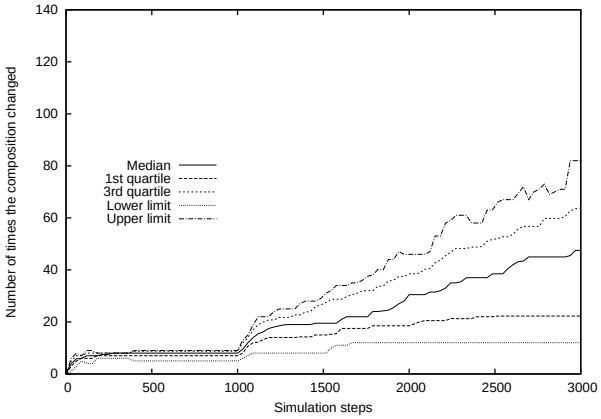


Figure 6. Stability of the service value chain with resource value as migration condition.

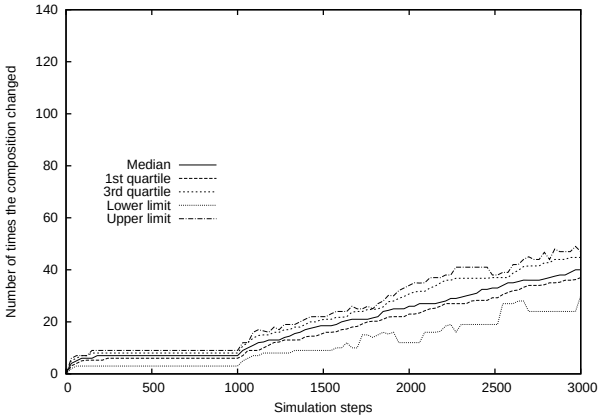


Figure 7. Stability of the service value chain with past interactions as migration condition.

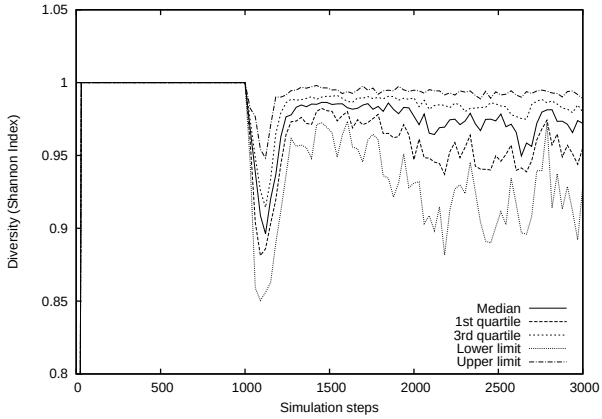


Figure 8. Ecosystem diversity across the environments with resource value as migration decision rule

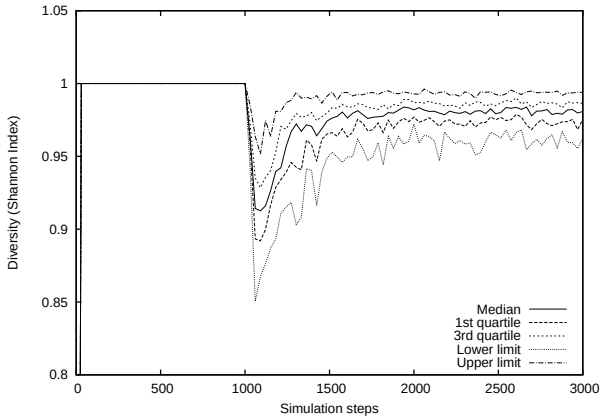


Figure 9. Ecosystem diversity across the environments with oast interactions as migration decision rule

time. This occurs due to the fact that an intelligent service with a low resource value will have a tendency to migrate more often than others, thus varying the diversity of environments and degrading the overall system. On the other hand, when past interactions is used as the migration condition (Figure 9), the distribution of diversity tends to stabilise above 0.95. This is due to the migration condition considering those intelligent services that are not that needed in an environment because they interact less. That is, the structure of the emerging compositions does not require those intelligent services. As a consequence, environment diversity is high and stable.

In summary, a distributed ecosystem environment enabling migration using past interactions as the migration condition brings the following benefits:

- **Improvement of service value chain.** Even when more dynamism is allowed due to environment distribution, the service value chain is increased.
- **Low variation of service value chains.** This renders this migration condition as an enabler of system robustness.
- **High diversity (evenness) of intelligent services across the environments.** This renders this migration condition as a good load balancer thus minimising system degrading.

V. Related Work

Service composition has seen many approaches proposing methodologies and techniques to compose atomic services into more complex ones, fulfilling the users' demand [16] and the ever growing complexity of services. Adaptive and self-organising techniques are emerging as valid approaches to achieve service composition. For example, in [17] a self-organising technique is used to combine service composition with the discovery process into one step. They use an agent based approach aided with contract net protocol to achieve service composition. Then Cloud participants and resources are mapped as agents which negotiate in order to compose the provided services. Additionally, a self-organising multi-agent system approach is used in [7], where agents seeking to be processed are able to search and dynamically organise themselves by composing the offered services, e.g., routing or processing, following an ant based self-organising mechanism. Moreover, [4] presents experiments using ecosystem in the context of manufacturing, where compositions of tasks to realise workflow processes are tested. Regardless of the focus on adaptation and/or self-organisation none of these approaches considers the heterogeneity aspect of intelligent services as our approach does.

In terms of experiments on ecosystems and technologies similar to services, DBE [18] is a platform for supporting business ecosystems. It considers a population of services which a genetic algorithm tries to find the optimal composition

 ISBN: 978-1-61208-391-9

with. Evolved service populations live in networked habitats in such a way that successful services tend to cluster where they are most required for compositions. Regardless of the ecosystem inspiration, this approach does not target heterogeneity of services nor adaptation and only focuses on composition optimisation. In another work, [19] presents a comparison of approaches on optimisation of service ecosystems in Cloud environments. The comparison is made using a process template and varying number of services which the approaches have to optimise. Although that work uses ecosystems as part of its context and uses two approaches for services, these approaches are never combined to encompass heterogeneity, a key aspect of intelligent services. In addition, [20] describes an ecosystem inspired architecture for supporting dynamic scenarios such as service ecosystems. The architecture considers entities such as flora and consumers, and tuple species representing niches by which such entities interact. The entities have needs and a "happiness" level they try to optimise by fulfilling their needs. Experiments show a balanced state of "happiness" levels across the predefined niches. However, these experiments only show their capacities of self-organisation and convergence to a solution. In contrast, our experiments not only cover the convergence to a solution, but also how migration keeps a balanced and diverse system without negatively affecting the quality of the solution.

VI. Conclusion

The contribution of this paper is an approach where ISBS are treated as ecosystem environments, which form a distributed ecosystem environment when they are interconnected to each other. Three elements of ecosystems are incorporated in our approach: food chains, environment, and migration. An implementation and subsequent experiments demonstrate the advantage of the approach for ISBS.

Our results demonstrate that implementing ISBS as ecosystems using migration of intelligent services based on past interactions, could bring a benefit to the system because a higher service value chain is obtained, which is desirable, when more dynamism is added to ISBS. Even when more dynamism is enabled, both a low variation (i.e., high stability) of composition of solutions and a high diversity of intelligent services across the ISBS are maintained, which minimise system degrading. Our results sustain the claim that ISBS implementing the ecosystem approach presented here can unlock the potential of intelligent services by enabling the adaptation of existing compositions while remaining a robust system. Future work along this line consists of implementing these features in an industrial setting, e.g., manufacturing.

References

[1] M. Wooldridge, An Introduction to Multi-Agent Systems. John Wiley & Sons, 2002.

[2] T. Erl, Service-oriented architecture: concepts, technology, and design. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference, 2005.

[3] C. A. Marín et al., "A Conceptual Architecture Based on Intelligent Services for Manufacturing Support Systems," in IEEE International Conference on Systems, Man, and Cybernetics. IEEE Computer Society, oct 2013, pp. 4749–4754.

[4] C. A. Marín, "Evaluation of an Ecosystem-Based Architecture for Adaptation to Cascades of Changes," in IEEE International Conference on Systems, Man, and Cybernetics. IEEE Computer Society, oct 2013, pp. 4760–4765.

[5] C. A. Marín et al., "Application of Intelligent Service Bus in a Ramp-up Production Context," in Proceedings of the Industrial Track of the Conference on Advanced Information Systems Engineering 2013 (CAiSE'13), CAiSE-IT 2013, V. Pelechano, G. Regev, and Y. Pigneur, Eds. CEUR Workshop Proceedings, aug 2013, pp. 33–40.

[6] J. Mendes, P. Leitão, F. Restivo, and A. Colombo, "Service-Oriented Agents for Collaborative Industrial Automation and Production Systems," in Holonic and Multi-Agent Systems for Manufacturing, ser. Lecture Notes in Computer Science, V. Mařík, T. Strasser, and A. Zoitl, Eds. Springer Berlin Heidelberg, 2009, vol. 5696, pp. 13–24.

[7] J. Barbosa and P. Leitao, "Enhancing service-oriented holonic multi-agent systems with self-organization," in International Conference on Industrial Engineering and Systems Management, Metz, 2011, pp. 1373–1381.

[8] D. A. Chappell, Enterprise service bus, 1st ed. Sebastopol, Calif: O'Reilly, 2004.

[9] C. A. Marín, I. Stalker, and N. Mehandjiev, "Engineering Business Ecosystems using Environment-Mediated Interactions," in Engineering Environment-Mediated Multi-Agent Systems: Selected, Revised and Invited Papers, ser. Lecture Notes in Artificial Intelligence (LNCS), D. Weyns, S. Brueckner, and Y. Demazeau, Eds. Heidelberg: Springer-Verlag, Jul. 2008, vol. 5049, pp. 240–258.

[10] J. Otsuka, "A Theoretical Characterization of Ecological Systems by Circular Flow of Materials," Ecological Complexity, vol. 1, no. 3, 2004, pp. 237–252.

[11] A. Rossberg, H. Matsuda, T. Amemiya, and K. Itoh, "An explanatory model for food-web structure and evolution," Ecological Complexity, vol. 2, no. 3, Sep. 2005, pp. 312–321.

[12] F. Lécué and N. Mehandjiev, "Seeking Quality of Web Service Composition in a Semantic Dimension," IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 6, jun 2011, pp. 942–959.

[13] G. A. Lopardo, J. de la Rosa i Esteva, and N. Hormazábal, "A Monitor for Digital Business Ecosystems," in Artificial Intelligence Research and Development, ser. Frontiers in Artificial Intelligence and Applications. Amsterdam: IOS Press, 2007, pp. 77—86.

[14] C. A. Marín, G. Lopardo, and N. Mehandjiev, "An Ecosystem-based Analysis of the Impact of an Interoperability Tool to a Network of SMEs," in 20th International Conference on Collaboration Technologies and Infrastructures, WETICE 2011. IEEE Computer Society, jun 2011.

[15] J. Barbosa and P. Leitao, "Simulation of multi-agent manufacturing systems using Agent-Based Modelling platforms," in International Conference on Industrial Informatics. IEEE, jul 2011, pp. 477–482.

[16] J. Rao and X. Su, "A Survey of Automated Web Service Composition Methods," in Semantic Web Services and Web Process Composition, ser. Lecture Notes in Computer Science, J. Cardoso and A. Sheth, Eds. Springer Berlin Heidelberg, 2005, vol. 3387, pp. 43–54.

[17] I. Al-Oqily and A. Karmouch, "A Decentralized Self-Organizing Service Composition for Autonomic Entities," ACM Transactions on Autonomous and Adaptive Systems, vol. 6, no. 1, Feb. 2011, pp. 1–18.

[18] G. Briscoe and P. De Wilde, "Digital Ecosystems: Evolving Service-Orientated Architectures," in BIONETICS'06: Proceedings of the 1st international conference on Bio inspired models of network, information and computing systems. New York: ACM Press, 2006.

[19] U. Wajid, C. A. Marín, and N. Mehandjiev, "Optimizing Service Ecosystems in the Cloud," in The Future Internet, ser. Lecture Notes in Computer Science (LNCS). Heidelberg: Springer-Verlag, may 2013, vol. 7858, pp. 115–126.

[20] C. Villalba, M. Mamei, and F. Zambonelli, "A self-organizing architecture for pervasive ecosystems," in Self-Organizing Architectures, ser. Lecture Notes in Computer Science. Heidelberg: Springer-Verlag, 2010, vol. 6090, pp. 275–300.

 ISBN: 978-1-61208-391-9

Cyber Organic System-Model – New Approach for Automotive System Design

Daniel Adam
BMW Forschung und Technik
80992 Munich, Germany
E-mail: Daniel.Adam@bmw.de

Joachim Fröschl
BMW AG
80788 Munich, Germany
E-mail: Joachim.Froeschl@bmw.de

Uwe Baumgarten
Technische Universität München
Fakultät für Informatik
Lehrstuhl/Fachgebiet für Betriebssysteme
85748 Garching, Germany
E-Mail: baumgaru@tum.de

Andreas Herkersdorf
Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Integrierte Systeme
80290 Munich, Germany
E-Mail: Herkersdorf@tum.de

Hans-Georg Herzog
Technische Universität München
Fakultät für Elektrotechnik und Informationstechnik
Fachgebiet Energiewandlungstechnik
80333 Munich, Germany
E-Mail: hg.herzog@tum.de

***Abstract*—Modern vehicles are characterized by multiple systems to represent customer functions. The steady increase of these functions leads to an increasing system complexity. This paper describes a structured approach to better master the complexity for future electric and electronic systems. The new approach is based on a combination of two existing system models. The first system model is based on a cybernetic perspective for a management approach for an electrical energy system. The second system model is based on the work in the section "Organic Computing" of the "Gesellschaft der Informatik e.V.". In addition, principles were taken from the human body to design this system approach. Based on these, the Cyber Organic System model is proposed for use in automotive E/E systems. Also, this model focuses the distribution of the overall function to the 3C (Car, Consumer Device, Cloud) locations. Furthermore, the use of this model in automotive software system design will be outlined on the basis of examples.**

***Index Terms*—Cyper Physical System; Electric and electronic vehicle architecture; Bio-inspired computing;**

I. Introduction

Currently, various approaches for the modeling of complex systems are discussed. In the context of a vehicle data networking in the sense of an overall network, the cybernetic model of fEPM (flexible Energy and Power Management) [1] and the Organic Computing (OC) [2] model appear most promising for automotive applications. Besides of the two mentioned models, there are further models. One of these models is the model of Deutsch, cited in Rittmann [3]. This model is characterized by the combination of regulatory and memory functions. On one hand, the fEPM model has already been tested in a vehicle with a high degree of maturity. On the other hand, the OC model has additional features. This paper discusses the combination of the two models for an overall approach for a bio-inspired software architecture for vehicles. Because the two models have a certain similarity, we discuss the combination of the two models into an overall approach in this paper.

This approach should enable the homogeneous cross linking between vehicles and the (surrounding) infrastructure. IoT (Internet of Things) leads to a continuous increase in the importance and relevance of cross linking. An important feature of the cross linking structure is the distributed function execution and the rising intelligence of these systems.

The rest of the paper is structured as follows. Section II presents the related work. Section III describes the mapping of the fEPM, OC model, and the COS model. Section IV gives an overview of different scenarios of the COS model in a vehicle. In Section V, discusses the applicability of COS in a vehicle. Section VI summarizes the paper.

II. Related Work

In this section, the two models are described, which were combined to the COS model in the following section. In addition, Table I shows other related models. The following models are related: VSM (Viable System Model) from Stafford Beer [4], NASREM (NASA/NBS Standard Reference Model for Telerobot Control System Architecture) from NASA [5], fEPM from Joachim Fröschl [1] and OC from the Organic Computing Initiative [2]. All of these models use the MAPE (Monitor, Analyze, Plan and Execute) or the SMPA (Sense, Model, Plan and Act) pattern [6], [7].

The related models don't consider about the current development, like IoT and Cloud. Therefore, the COS model is a consequent enhancement under consideration of requirements of the current developments.

A. fEPM - flexible Energy and Power Management

The fEPM [8] is based on a recursive application of the following drafted cybernetic basic model, as shown in Figure 1. This basic model is based on the VSM (Viable System Model) from Stafford Beer [4]. It features 5 system levels. The system level 1 is mostly defined by system values. Beside

 ISBN: 978-1-61208-391-9

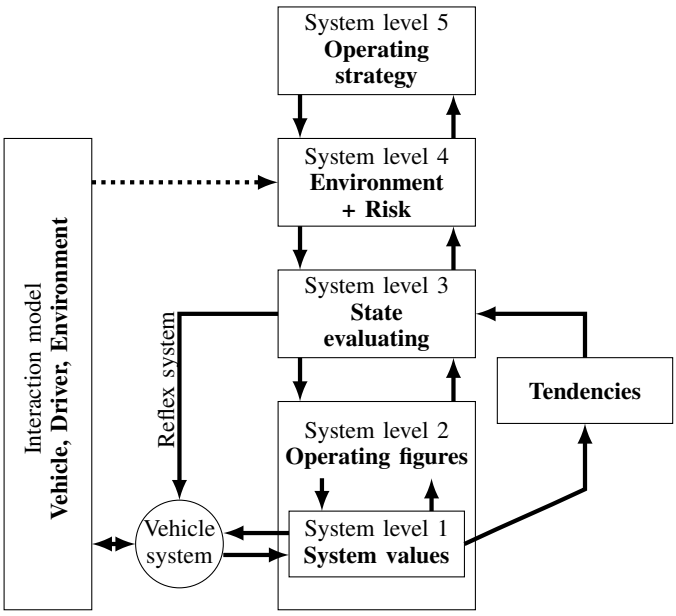


Fig. 1. The fEPM model.

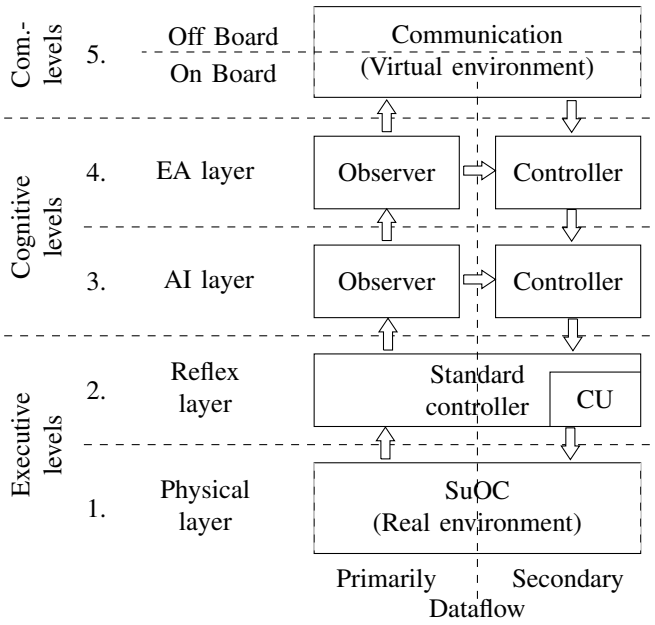


Fig. 2. Reference Model of the OC model (Source: [2]).

TABLE I. OVERVIEW OF THE RELATED MODELS

	VSM	NASREM	fEPM	OC
Layer #	5	6	5	5
Time of origin	1972	1985	2006	2010
Main characteristic	First biological model	2D Layer model	Technical transformation of the VSM	Learning ability

the physical connections, control and steering functions are included.

The system level 2 condense the system values into operating figures.

The system level 3 determines the operating figures and tendencies that are the analysis of the variation in time of the system values and the operating figures, with deposited knowledge into system states. This level contains also the autonomous, state based system modifications for the purpose of system stabilization.

The system level 4 combines the internal system states with the external system states based on the environment information under observation of the risk; although the modification of a higher hierarchy level is included in the same way.

The coupling of the environment information is filtering the relevant information out of the system specific environment.

The system level 5 contains the operating strategy, which defines the conscious behavior. In this level, the regulation values, also known as modificators, are composed [9], [10].

B. OC - Organic Computing

Figure 2 illustrates the OC model, which consists out of five layers. The lowest layer is the physical layer, which contains the environment, the actuators, and sensors. The second layer is the reflex layer, a kind of protection layer to ensure fast and correct reaction. The third layer is an AI (Artificial intelligence) layer, which is responsible for quick responses to unfamiliar situations. The fourth layer is the EA (Evolutionary Algorithm) layer to generate long term strategies by using evolutionary approaches - selection, mutation and recombination, - and simulation. The fifth and topmost layer is used to communicate with other components. This communication happens mainly through different (data) models (environmental, vehicle, and driver model), which enables modeling the real world abstract. This layer contains an image of the lowest layer of [2].

III. MAPPING AND DESCRIPTION OF THE COS MODEL

In this section, the identified modules from the fEPM and OC models are explained. This identification of the modules is the necessary basis for mapping these two models. A fundamental fact is to divide the modules to related modules with similar behavior and properties. This implies that any property or object is not included in both approaches. Functional properties of the two models are targeted to combine so that a new model is created with a bigger functional scope. Figure 3 illustrates the mapping of each module. For traceability reasons, each module has its own identifier. We introduce these identifiers in the description of the modules (see subsection III-A), which will be used in the description section of the COS (see subsection III-B).

A. Modules

Consecutively, the particular modules are compared in detail, to define the COS model afterwards.

 ISBN: 978-1-61208-391-9

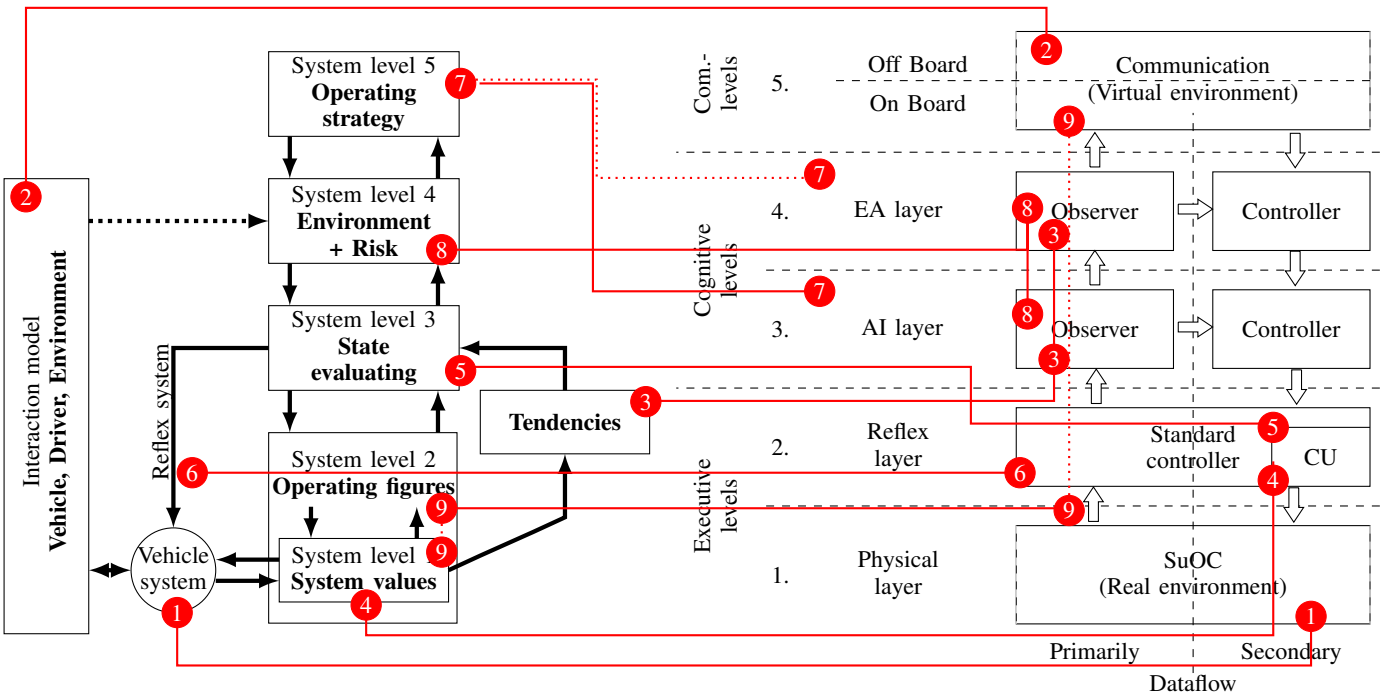


Fig. 3. Mapping between the OC and the fEPM model

1) Sensors (1a): This module includes in both approaches the necessary sensors or other general information sources that are necessary for the investigation of relevant data or system values.

2) Actuators (1b): This module includes in both approaches the necessary actuators and general actuator elements that are necessary for the relevant system modifications.

3) Environment (2): With the environment, all influences from outside the system border are referred. This means on one hand the acquisition and on the other hand the impact of the environment. In both approaches, a real and virtual environment can be distinguished.

a) Real environment (2r): Within the real environment the physical objects are included.

b) Virtual environment (2v): Within the virtual environment instances respective models of the relevant physical objects from the real environment are necessary for the corresponding functions. Because the fEPM makes no distinction between a virtual and a real environment, such differentiation is implemented into the OC module.

4) Data analysis (3a-c): This means that the internal system values should be analyzed. Depending on the system level, the abstraction level changes and a different view on data analysis and data calculation is necessary.

5) Limitation (physical) (4): To avoid data overflow within the modification the so called modificators are limited in the first step of propagation.

6) State evaluating (reflex) (5): Within this level, an assessment of the system values, operating figures, and tendencies is done. If these parameters are out of a defined range, taking care this layer to use values these are within the range of values. The allowed range is defined by tolerability borders wherein the system is able to fulfill explicit operating strategies. If the values are moving out of the defined range, a system reaction has to be initiated like a reflex for stabilizing itself. In a human organism this behaviour is called homeostasis [11].

7) Reflex (execution) (6): To stabilize the system, immediate and direct reactions are necessary. In addition, the operating strategy defines the self-awareness behavior, where some actions must be blocked.

8) Conscious behavior (7): When the system is in a stable state, the system works with a conscious behavior, for example as an operating strategy. While in the fEPM model only one level of conscious behavior is implemented, in the OC-model there are two levels that exist. The lower AI level is conform to the system level 5 of the fEPM model. The upper EA level contains additional evolutionary algorithms to construct a long range operating strategy performance result. Therefore, a higher degree of learning aptitude and self-dependence within the functionality should be enabled.

9) Aggregation of data (8a-c): The accumulated data are collected for an increase of information entropy. The difference in a-c is done in the same matter and argumentation is done similarly in data analysis.

10) Recursivity: Both approaches can be used in a recursive way to reduce complexity. With the use of recursion, the higher

 ISBN: 978-1-61208-391-9

instances are unloaded and the lower instances are reinforced to their self-dependence. The lower instances have their own duties and freedom of action. This autonomy is based on the principle of subsidiarity and enables a federal distribution of functionality. This enables a higher capacity of action shown in the example of the OC model in Figure 5.

B. Cyber Organic System Model

The discussed modules can be combined to the cyber organic system model, called COS model represented in Figure 4. Therefore, the single key features of the identified levels are drafted and dedicated to the modules consecutively.

Therefore, it is possible that the left and the right side of the model can be implemented on different Electronic Control Units (ECUs). It is also possible to omit some levels of the model.

1) Communication level: Here, the instructions to the system behavior are partially calculated from modules of a higher hierarchy level and partially from calculated modules of the same hierarchy level received. Furthermore, calculated data with partially higher data entropy is delivered. The different communication partners could stay within the vehicle (on board) or outside the vehicle (off board).
Also, the modules 2v, 3c, 8c are included.

2) Intelligence level: This level includes the specific data processing within the observer unit and a simulation unit within the control unit to learn a long term operating strategy. An additional validation unit is necessary to validate the data and information, which are developed on non-functional verified components.
Also, the modules 2v, 8b are included.

3) Strategy level: Within this level, the learned operating strategy from the intelligence level is used to execute a conscious behavior in a fast and optimal way. Therefore, the necessary information is processed in the observer unit demanded by the control unit.
Also, the modules 2v, 8a are included.

4) Reflex level: In this level, a first and fast analysis in the observe unit (OU) is executed. If necessary, a reflex reaction is immediately initiated. Similar to the human nerve system the conscious behavior is blocked or overruled within the control unit (CU). The human nervous system is spoken of a inhibitory interneuron [12]. Thus, effects self-awarness, - self-protection, and self-stabilization - of the system. In opposite to both higher levels, – intelligence and strategic level – fixed reflexes are used based on an identical data acquisition. Therefore, the freedom of action and possibilities of calculation are limited.
Also, modules 5 and 6 are included.

5) Objects layer: Within this level, the data sources and drains of the function model are implemented. On one hand a real hardware for sensors and actuators can be used. On the other hand further software subsystems (e.g. COS stacks) can be embedded.
Also, modules 1a, 1b, 2r and 4 are included.

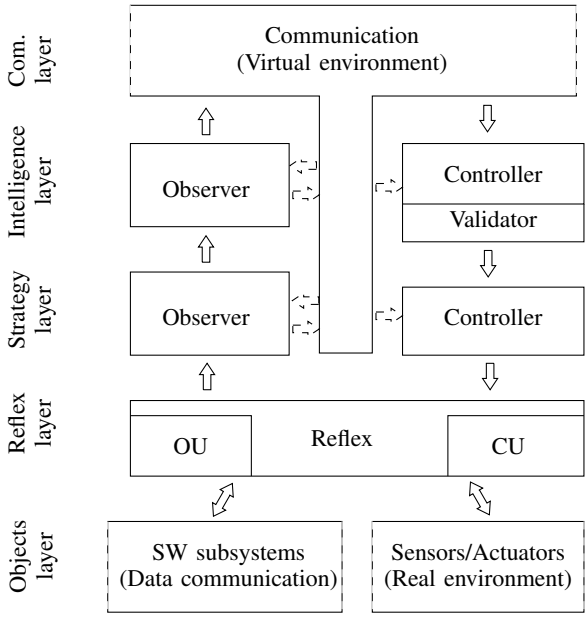


Fig. 4. Structure of the Cyber Organic System model.

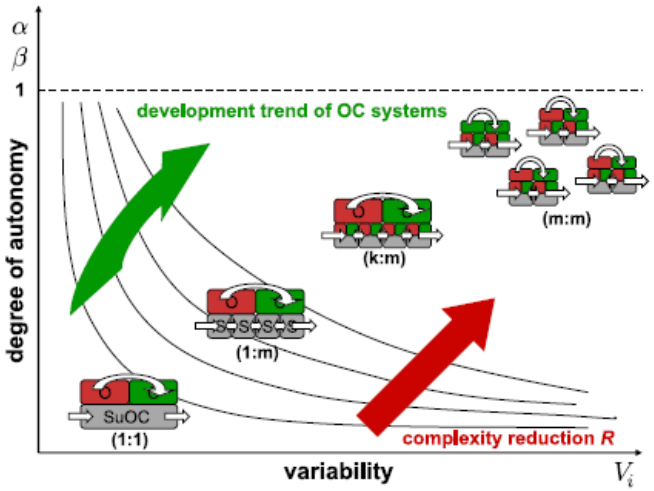


Fig. 5. Degree of variability and autonomy of a OC model in dependence on the system topology (Source: [2]).

IV. The Cyber Organic System Model in Vehicle

On one hand, the consideration of a function with a COS model is possible. On the other hand using the recursion principle a complete system with many different functions is also possible. First, the complexity of the entire system is reduced. Second, the degree of autonomy and the variability is increased (see Figure 5) [2].

The layered architecture of the COS model enables a flexible and temporary relocation of (partial) functionalities or groups towards further locations (3C).

In the actual hierarchy formation several approaches are possible: a centralized and decentralized approach. This is application may be dependent on not only the platform or model, but even on the equipment of actual vehicle.

 ISBN: 978-1-61208-391-9

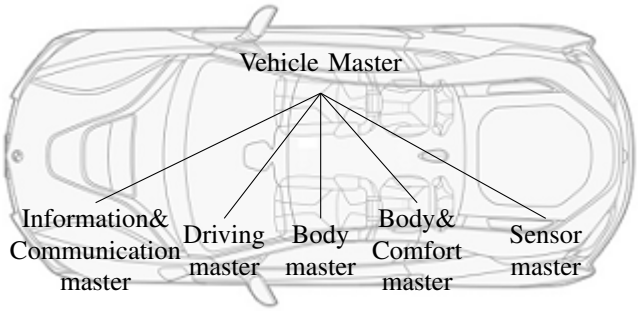


Fig. 6. Central approach in the full vehicle with the COS model.

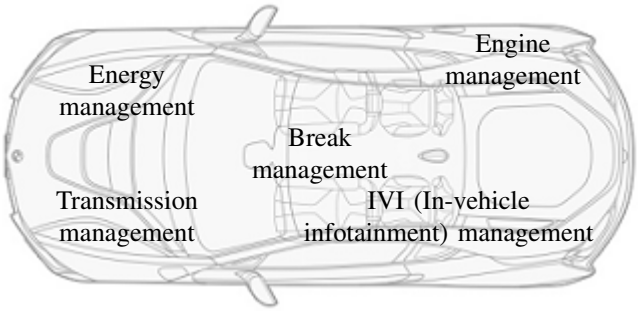


Fig. 7. Decentral approach in the full vehicle with the COS model.

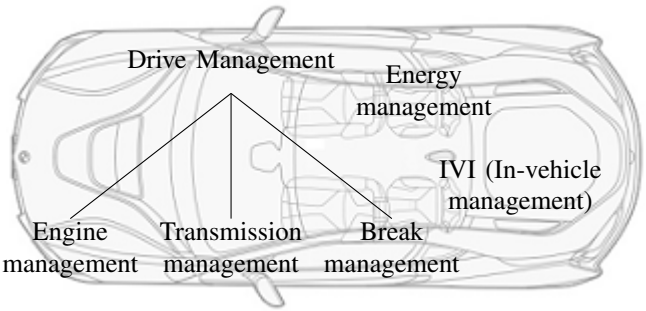


Fig. 8. Hybrid approach in the full vehicle with the COS model.

In the centralized approach, there is a vehicle master at the top of the hierarchy and each function is arranged directly or indirectly under this master. In the decentralized approach, independent groups of functions are set up, which also have recursive structures. The extreme case is only having one function in the group. A division of the functions is possible in either domain, see Doman Controller approach according [13], or functional point of view. The size of the groups probably depends on the functions and domain. Different behaviors already exist between domains, as an example, the scaling functions mentioned. With scaling, the functional scope changes. This is reasonable, because of the larger range of functions in the vehicle. A second reason is that existing functionality is being extended. Therefore, their functionality is growing. Which, in turn, results in changes of the partitioning. For example, the light control can be considered. A standard front light requires a smaller software scope, as a xenon or laser light. Another example of the equipment variant can be made the variety of a seat. From pure mechanisms over an electrically adjustable up, to a fully air-conditioned seat with a massage function. While the functions are partitioned at a high scale, a plurality of control devices in a domain matching in another domain are the same case, a high integration strategy is followed. In the case of high integration, several (independent) functions will be combined on one electronic control unit [14].

Figures 6, 7 and 8 outline the different organization options within a vehicle. The topmost element of each organization structure is the coordinator of the underlying elements and the communication partner to another vehicle or cloud. The organization structure depends on the configuration of every vehicle. Probably, neither of the two extreme approaches will be usedInstead, a hybrid approach between both, which frequently occurs in distributed systems, will be used.

V. Discussion

When the two approaches are compared, it indicates that the fEPM approach has a high similarity with the OC approach, as listed in Table II.

The fEPM model follows a single-generation and an individual approach. The OC model follows a multi-generational approach. This difference is given by the fact that the OC approach has besides the strategy an intelligence layer to generate completely new strategy plans through evolutionary algorithm. Also, the fEPM works only with two (online) timeframes. In contrast, the OC approach has three timeframes. In addition to the two online timeframes, an offline timeframe exists, which generates completely new strategy plans. This results in a self-development of the strategy. This leads to a continuous system improvement through targeted objective adjustments and optimization.

TABLE II. TAKS AND FUNCTIONS FOR DATA DISTRIBUTION AND PROCESSING

fEPM	**OC**
1 generation and 1 individual	Multi-generational approach and mutli individuals
Operating strategy is in the system level 5	Operating strategy is in the AI layer
Validation is doing in the switch and limit unit	
Strategy adaptation	Strategy generation
2 Timeframes •Online (short term/Reflex) (S3) •Online (medium term/Awareness) (S5)	3 Timeframes •Online (short term/Reflex) (Reflex layer) •Online (medium term/Awareness) (AI layer) •Offline (long term/Self generation of new strategies) (EA layer)

Furthermore, the modular structure of the COS model allows transparent segmentation for the actual function. This enables a flexible partitioning to various control devices or other locations - the 3C locations. This flexibility can not only be used in the development, but also during the operation of a temporary relocation of individual layers is possible. Table III shows a possible distribution of the different layers at the 3C locations. The decisive factors are from today's per-

 ISBN: 978-1-61208-391-9

spective: communication times, deadlines, energy consumption, safety requirements (Verification and certification of the components), and the necessary computing power. In order to fulfill the requirements of the ISO 26262 (Road vehicles - Functional safety), approaches or related norms from other industries, for example, from the railway and chemical industry, can be applied. The following norms should be mentioned: EN 50159 (Railway applications - Communication, signaling and processing systems), IEC 61508 (Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems) or IEC 62443 (Industrial communication networks - Network and system security). Through these approaches, SIL 3 (Safety Integrity Level) requirements can be fulfilled or made higher. This corresponds to the highest ASIL (Automotive Safety Integrity Level) [15]. This is taken into account the Validator in the intelligence layer.

Because of the software scopes (Lines of code) and the required computing power, the necessary functional validation for the simulation, the intelligence layer is not realizable for all of its functions. In order to ensure operational safety, the COS model contains a Validator. The Validator is used to verify the results from the simulation of a non-functional or unsafe area to use in a functional protected area according ISO 26262.

TABLE III. POSSIBLE PARTITIONING OF THE INDIVIDUAL LAYERS ON THE 3C LOCATIONS

Layer	Car	Consumer Device	Cloud
Reflex	X	(X)	
Strategy	X	X	X
Intelligence	(X)	X	X

Legend:
X = Without any restrictions (X) = Possible with restrictions

VI. SUMMARY

The resemblance of the two models - fEPM and OC - allows a merge into the COS model, which combines the properties of the two models. Due to the same objects and features of the models from the two approaches, the unification could be done with little effort. In addition to the outline of the two models, there was a modularized representation of COS models and a final exemplary application for partitioning functionality in a vehicle.

Furthermore, it was already the (detailed) implementation and use considered, there were many analogies to the functioning of the human body. Also, the CU unit came in consideration the principles of the human body in the COS model. As added value of the combination and the analogies we expected, a revolution of further development of the vehicles system to something like humans, is considered the highest stage of evolution.

The future work should be an implementation and an evaluation of the COS model within a vehicle architecture. The first step is the implementation of selected functions. The second step is the implementation of a representative set of vehicle functions. As a basis for the implementation the results of the development of the fEPM model can be used.

REFERENCES

[1] J. Fröschl and H.-G. Herzog, "A new kind of an Energy Management System," Bad Boll, April 2015.

[2] C. Mueller-Schloer, H. Schmeck, and T. Ungerer, Organic Computing - A Paradigm Shift for Complex Systems. Springer, Apr. 2011, ISBN: 978-3-0348-0130-0.

[3] G. Rittmann, Der Umgang mit Komplexitt: Soziologische, politische, konomische und ingenieurwissenschaftliche Vorgehensweisen in vergleichender systemtheoretischer Analyse. Nomos, ISBN: 9-783-8487-0990-8.

[4] S. Beer, Brain of the Firm. Chichester, England; New York: John Wiley & Sons, Jun. 1995, ISBN: 978-0-4719-4839-1.

[5] J. S. Albus, H. G. McCain, and R. Lumia, "NASA/NBS standard reference model for telerobot control system architecture (NASREM)," NBS/NIST Technical Notes, no. 1235, 1987, pp. 1 – 98, Last Accessed: 2015-02-15. [Online]. Available: http://archive.org/details/nasanbsstandardr1235albu

[6] IBM, "An Architectural Blueprint for Autonomic Computing," IBM, Tech. Rep., Jun. 2005, Last Accessed: 2015-02-15. [Online]. Available: http://www-01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf

[7] J. Hertzberg, H. Jaeger, U. Zimmer, and P. Morignot, "A framework for plan execution in behavior-based robots," in Intelligent Control (ISIC), 1998. Held jointly with IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA), Intelligent Systems and Semiotics (ISAS), Proceedings. IEEE, Sep. 1998, pp. 8–13, ISBN: 0-7803-4423-5.

[8] T. P. Kohler, J. Fröschl, and H.-G. Herzog, Systemansatz fur ein hierarchisches, umweltgekoppeltes Powermanagement, C. Hoff and O. Sirch, Eds. Renningen: expert Verlag, ISBN: 9-783-8169-3010-5.

[9] T. Kohler, J. Fröschl, and H.-G. Herzog, "Systemansatz für ein hierarchisches, umweltgekoppeltes Powermanagement," in Elektrik/Elektronik in Hybrid- und Elektrofahrzeugen, O. Sirch, Ed. Haus der Technik e.V., 2010, ISBN: 9-783-8169-3010-5.

[10] T. P. Kohler, Prädiktives Leistungsmanagement in Fahrzeugbordnetzen. Berlin Heidelberg New York: Springer, 2014, ISBN: 978-3-6580-5012-2.

[11] A. Faller and M. Schuenke, The Human Body: An Introduction to Structure and Function, 1st ed. Georg Thieme Verlag, April 2004, ISBN: 9-781-5889-0122-4.

[12] J. S. Schwegler and R. Lucius, Der Mensch - Anatomie und Physiologie. Georg Thieme Verlag, Jul. 2011, ISBN: 9-783-1316-8935-1.

[13] H. Krimmel, H. Deiss, W. Runge, and H. Schürr, "Electronic networking of driveline and chassis," ATZ worldwide, vol. 108, no. 5, 2006, pp. 5–8, ISSN: 2192-9075.

[14] D. Adam, S. Tverdyshev, C. Rolfes, T. Sandmann, S. Baehr, O. Sander, U. Baumgarten, and J. Becker, "Two Architecture Approaches for MILS Systems in Mobility Domains (Automobile, Railway and Avionik)," in International Workshop on MILS: Architecture and Assurance for Secure Systems, Amsterdam, Jan. 2015.

[15] R. Bris, C. G. Soares, and S. Martorell, Reliability, Risk, and Safety, Three Volume Set: Theory and Applications. CRC Press, 2009, ISBN: 9-780-2038-5975-9.

How Adaptation and Transformation Complement Each Other to Potentially Overcome Signature Mismatches on Object Data Types on the Basis of Test-Cases

Dominic Seiffert

Software Engineering Group
University of Mannheim
Mannheim, Germany
Email: seiffert@informatik.uni-mannheim.de

Oliver Hummel

Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: hummel@kit.edu

***Abstract*—The challenge of providing fully automated adaptation is tackled by many approaches in literature. Thereby, the class of signature mismatches presents the challenge of matching object data types that provide the same semantics but are syntactically incompatible. We explain in this paper how adaptation, complemented by transformation, can potentially solve the problem on the basis of test cases in the object-oriented world.**

Keywords–automated adaptation; signature mismatches; object data types; test cases.

I. INTRODUCTION

Software building blocks such as objects need to connect their interfaces in order to create new functionality. Unfortunately, this connection is not always possible because of signature mismatches. A simple example for a signature mismatch occurs, when the interfaces to connect have different names. A more challenging task is a signature mismatch for deviating parameter or return object data types. For object data types a subclass instance can be delivered when a super class instance is expected, according to Liskov Substitution Principle [2]. This is not possible, however, when the subclass relationship does not exist, even for a semantically equal instance of a different type.

Signature mismatches are tackled by many approaches in literature, towards the goal of providing fully automated adaptation. Thereby, an adapter gets interposed which handles the message wiring between the involved software building blocks. However, current approaches from the object-oriented communities lack the ability to overcome signature mismatches on object data types that are syntactically incompatible but provide the same semantics. We believe that it is a need to challenge this task in order to further improve fully automated adaptation.

In the remainder of this paper, we propose adaptation complemented by transformation, on the basis of test cases, as the possible solution on overcoming signature mismatches for object data types in object-oriented programming. The necessary background information on adaptation and a more detailed description of the problem is illustrated in Section 2. In Section 3 the solution is proposed. Section 4 refers to related work. Section 5 states the conclusion.

II. BACKGROUND

Signature mismatches can potentially be solved by an adapter that gets interposed between the software building blocks [3]. The Gang of Four Object Adapter pattern [4] provides a well-known solution in the world of object-oriented programming. The following adaptation example in Figure 1 explains the pattern in more detail: Let A and B be two simple object data types that are not connected by hierarchy, and simply provide each the methods set(int) and get():int, to set and retrieve an integer value. Let further be StackA and StackB two simple stack implementations, which are also not connected by type hierarchy. An instance of StackA allows an instance of A to be pushed on and popped off, whereas StackB allows the same for an instance of B. The Client depends on the StackA interface but wishes to use the methods push(B) and pop():B from the adaptee StackB. The StackA interface provides the methods push(A) and pop():A that use the parameter and return type A. Therefore, a signature mismatch on object data types occurs for A and B.

In an idealized scenario, the Adapter implements the StackA interface and wraps the adaptee. The adaptee receives forwarded messages from the Client via the Adapter. More exactly, the Client invokes the set(A) method and the Adapter forwards the messages to the set(B) method of the adaptee StackB. The same happens vice versa for the pop methods and their return values.

In order to support a fully automated process, Hummel and Atkinson [7] proposed the idea of specifying the expected adapter's functionality in a test case that is used then during the adaptation process for checking the semantics of the adaptee. Figure 2 provides a sample test case for the previous example, where the client expects an adapter StackAB to be created that adapts StackA on StackB. This is specified in line no. 1 by the import statement. The expected functionality is simple: The object data type A gets instantiated and the value 5 is set to it in line no. 8. This instance is pushed on the adapter StackAB in line no. 9. In line no. 10 the pop() method gets invoked, which is expected to deliver an instance of A again. An invocation of get() on this instance is expected to deliver the value 5. This is required in order to pass the test as specified by the assertEquals statement in line no. 11.

Such a unit test case and a candidate to adapt serves as an input for the adapter generation tool [8] developed by Seiffert and Hummel [9]. The tool generates adapters that are based on the Managed Adapter Pattern [5], which is based on Fowler's identity pattern [6, p. 195]. The Managed Adapter Pattern solves the problem of the Gang of Four Object

 ISBN: 978-1-61208-391-9

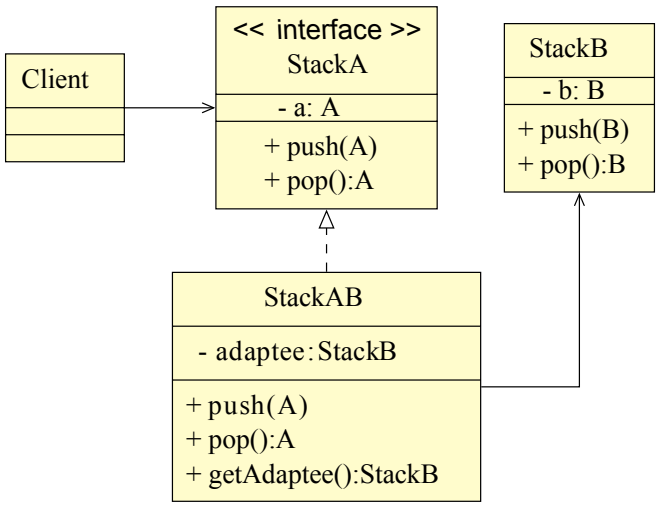


Figure 1. Adapter StackAB where StackB is the adaptee.

```
1 import adapter.StackAB;
2
3 public class TestCase extends junit.framework.TestCase {
4
5   public void test {
6    StackAB adapter = new StackAB();
7    A a = new A();
8    a.set(5);
9    adapter.push(a);
10   A a = adapter.pop();
11  assertEquals(5, adapter.get());
12  }
13 }
```

Figure 2. Test case where StackA adapts StackB.

Adapter Pattern when the adaptee expects its own type as a parameter argument or delivers it as a return type. The tool further overcomes signature mismatches on primitive data types, parameter permutations and a subset of object data types, namely arrays and collections [10] that share common semantics.

In the following example the situation has changed, as illustrated by UML diagram in Figure 3: Let the Client depend on StackOwnerA and let StackOwnerB play the role of the adaptee. The Client wants to adapt StackOwnerA to StackOwnerB, which requires that the set(StackA) method must be matched on the set(StackB) method, and the get():StackA method must be matched on the get():StackB method. Therefore, a signature mismatch on object data types occurs for StackA and StackB. A potential solution on this problem is proposed in the following section.

III. Adaptation and Transformation on the Basis of Test Cases

When the client delivers the StackA instance to the adapter StackOwnerAB, by invoking the set(StackA) method, the first idea is to simply reuse the adapter AdapterBA from the previous example, as illustrated by Figure 4. Reusing an adapter requires the existence of an adapter repository, which has already been proposed by Gschwind [11].

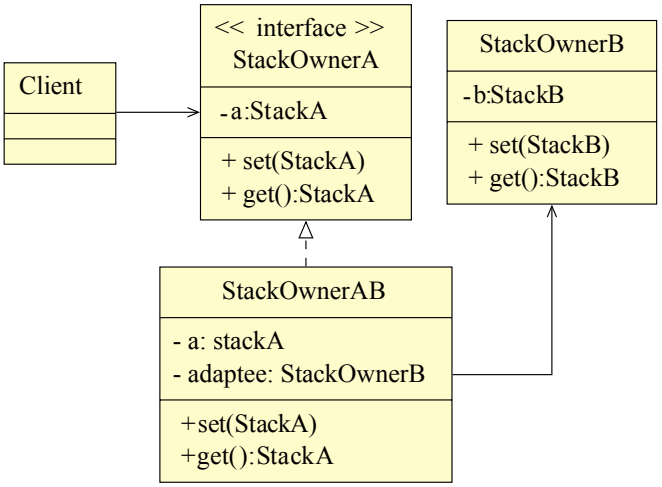


Figure 3. Adapter StackOwnerAB where StackOwnerB is the Adaptee.

```
1 StackOwnerAB {
2 set(StackA stackA) {
3  StackOwnerB adaptee = new StackOwnerB();
4  StackBA reusedAdapter = null;
5  If(repository.entryExists(adaptee, stackA.getClass())){
6   reusedAdapter = repository.getEntry(adaptee, stackA.
        getClass()));
7  }else{
8    reusedAdapter = generateAdapter(adaptee, stackA.
        getClass());
9    repository.setMapping(stackA, reusedAdapter);
10 }
11 reusedAdapter.setAdaptee(stackA);
12 adaptee.set(reusedAdapter); //forwarded.
13 }
14 }
15 Repository {
16  ...
17  public Object generateAdapter(Class from, Class to){
18   //here the adaptation happens
19  }
20 }
```

Figure 4. Reusing an existing adapter.

More precisely, in line no. 5 the repository is checked whether an adapter StackBA already exists for the arriving instance stackA. If the repository cannot find an adapter StackBA, a new adapter StackBA must get generated, as indicated by line no. 17, and set to the repository. Nasehi and Maurer [12] propose that test cases should be equipped with a standard API, which we believe would support our adapter generation tool, because it takes takes test cases as an input. However, if test cases are not provided, the idea is to generate them automatically, which is not a futuristic scenario. For example, Galler and Aichernig [13] provide an overview on current test case generation tools, where some tools seem to be practical.

In line no. 11, the arriving StackA instance needs to be set as the adaptee for the adapter StackOwnerAB. This is different to a ”regular” adapter that usually creates its adaptee instance itself.

The adapter StackBA is assumingly not a subclass of StackB. Therefore, StackBA and StackB are mismatching object data types, and no instance of StackBA can be delivered

 ISBN: 978-1-61208-391-9

```
1 import adapter.StackOwnerAB;
2
3 public class TestCase extends junit.framework.TestCase {
4
5   public void test {
6    StackOwnerAB adapter = new StackOwnerAB();
7    A a = new A();
8    a.set(5);
9    StackA stackA = new StackA();
10   stackA.push(a);
11   adapter.set(stackA); //forward.
12   ...
13  }
14 }
```

Figure 5. Test case snippet where StackOwnerA adapts StackOwnerB.

```
1 StackOwnerAB {
2 set(StackA stackA){
3   StackB stack = transform(stackA);
4   StackOwnerB adaptee = new StackOwnerB();
5   adaptee.set(stack);
6  }
7 }
```

Figure 6. Adapter StackOwnerAB uses transformation.

```
1 Transformer.transform(StackA stackA):StackB {
2   StackB stackB = new StackB();
3   stackB.push(stackA.pop());
4   return stackB;
5 }
```

Figure 7. A simple transformation example.

when an instance of StackB is expected, as intended by line no. 12. To realize this, some cases must be considered, before the following background:

- A class declared as final can not be sub-classed.
- When the expected type is an interface type any instance of a class, which implements this interface, may be provided.

Therefore, the following cases are possible:

1) If StackB is a class type declared as final, then the adapter StackBA is not able to subclass it. Therefore, it cannot be forwarded, which is a severe limitation of this approach and calls for a more generally usable solution.
2) If StackB is a class type not declared as final, the adapter can subclass StackB. Therefore, it can be forwarded.

For the first case that StackBA cannot subclass StackB or StackB is not declared as an interface type, the following two potential solutions exist: First, if we would know the methods and their parameter values which were invoked on the arriving instance of StackA, i.e. the protocol, we could reuse the adapter StackAB. This is realized by reusing the protocol of instance stackA, in order to invoke the same methods with the same values on the adapter StackAB. The adapter StackAB provides the getAdaptee():StackB method as indicated by Figure 1. Therefore, by calling stackAB.getAdaptee(), the adapter StackAB returns an instance of StackB, which can be forwarded to the set(StackB) method of StackOwnerB. This idea requires that instances are monitored and that this information could be retrieved accordingly by a protocol. The idea of encapsulating such protocol information is proposed by Pintando [14] by using ”gluons”. These are special objects embedding interaction protocols between components. We believe that this protocol information could also be provided by a test case. For example, the test case snippet in figure 5, where StackOwnerA wants to adapt StackOwnerB, provides from line no. 7 to line no. 10 the protocol information that an instance of A is set a value of 5 and that this information is pushed on a StackA instance. Therefore, the idea is, to extract the protocol information from the test case.

The second solution to this problem is, in our believe, to provide a transformation mechanism. Transformation transforms ”state”. This means transformation extracts state from one instance and sets it to another instance. Thereby, the instances are not necessarily of a different type. The instances can mismatch by their state only. For example, an instance of StackA with the values 1,2,3 pushed on it, is different to another instance of StackA keeping the values 3,2,1. Compared to adaptation transformation requires different matchings for the methods, parameters and return types. Figure 6 and Figure 7 show how the adapter StackOwnerAB would implement a transformation mechanism, to transform the state of an instance of StackA to an instance of StackB. The transform method is assumingly provided by a class Transformer.

The challenge for this transformation is that the output parameter of the pop method, from StackA, is matched on the input parameter of the push method, from StackB, which is already challenged by the web service community ([15] [16]). But transformation, in the context of adaptation, does not mean that output parameters of methods are matched on input parameters of other methods only. Examples can be more challenging, as an adapter potentially needs to involve functionality from other classes, which are not necessarily provided as potential adaptees, in order to provide the expected functionality. For illustration purpose, let the Client expect the standard deviation to be calculated for the values he puts on the stack. This functionality needs to be assembled by the adapter. The adapter thereby works more like a facade [4]. Again, the client specifies the expected functionality by a test case in the first step. Thereby, he adds a specific comment to it, as shown by Figure 8 in line no. 10. This comment states that he expects the standard deviation to be calculated. The idea is, that this comment gets extracted out of the test case. Data mining tools could be used then, to determine that the standard deviation should be calculated. The standard deviation requires the calculation of the mean and variance. This knowledge could be retrieved from an ontology for instance. The information gets provided to the adapter. Figure 9 shows the potential transformation function that takes an array specified as *functionality* as an input. This array is filled with the content *mean*, *variance* and *deviation*, representing the information retrieved from the comment and ontology. Therefore, the adapter has to involve other ”helper”-classes, such as Mean, Variance and StandardDeviation, which it retrieves from the repository. Instances of these classes can either be already existing adapters or generated adapters. The latter are generated on the basis of equipped test cases or

```
1 import adapter.StackAB;
2
3 public class TestCase extends junit.framework.TestCase {
4
5   public void test {
6    Adapter adapter = new Adapter();
7   adapter.push(10);
8   adapter.push(5);
9   adapter.push(9);
10  //I want the standard deviation to be calculated.
11  assertEquals(2.16f, adapter.pop());
12  }
13 }
```

Figure 8. Test case enriched by comment.

```
1  Transformer.transform(StackA stackA, String[]
       functionality) {
2    Adapter mean = repository.getClass(functionality[0]);
3    Adapter variance = repository.getClass(functionality
         [1]);
4    Adapter standardDev = repository.getClass(
         functionality[2]);
5   StackB StackB = new StackB();
6        //calculate Mean
7   int mean = mean.calculate(values);
8      //calculate Variance
9    int variance = variance.calculate(mean);
10      //calculate StandardDeviation
11  int result = standardDev.calculate(variance);
12   StackB.push(result);
13   return StackB;
14  }
```

Figure 9. Transformation from StackA to StackB.

generated test cases.

IV. Related Work

The Morabit approach presented by Brenner et al. [17] is based on a prototype component framework that uses built-in tests. However, it does not consider the possibility of automated test case generation. The Java Object Instrumentation Environment, named JOIE, is one of the early adaptation approaches proposed by Cohen et al. [18]. JOIE allows the transformation of Java classes through byte code modification. For example, it allows the insertion of new code into the class to be modified and the change of data types or method names. Another approach proposed by Reiss [19] modifies the adaptees on the source code level to meet the expected requirements made by the Client. For such an invasive modification license problems can be problematic [20]. But we believe that it becomes necessary for adaptation on a deeper nested level. For example, let us assume that the content of the stackB instance, arriving at the set(StackB) method of StackOwnerB, should be displayed by a class Display which offers the methods show(OtherStack). The invocation of these methods should be inserted into the set(StackB) method provided by StackOwnerB either by source-code or byte-code modification. This requires that also the adapter generation of the adapter that adapts StackB to OtherStack gets inserted invasively. Kell [21] provides a rule-based approach named Cake which allows the transformation of object data types, but it requires the user writing mapping rules. We believe that the possible mappings should be detected and verified automatically during the adaptation process. The problem of service adaptation is already challenged by the web-service community by orchestration. For example, Eslmamichalandar et al. [22] provide an overview on web-service adaptation approaches. But approaches in the web-service community rely on xml-schema matching or ontology based reasoning. This is appropriate for syntactic matching, but it does not solve the problem in the object-oriented world when two syntactically incompatible but semantically equal type instances need to be adapted and potentially transformed. Gschwind [11] proposes the idea of an adapter repository where adapters can be retrieved by some meta information. Non existing adapters need to be provided by the client. The idea of creating them automatically is not considered.

V. Conclusion and Future Work

In this paper, we have provided potential solutions on overcoming signature mismatches on object data types on the basis of test cases. The potential solutions are adaptation and transformation. Transformation complements adaptation but works differently, as it extracts the state of an instance and sets it to another, and thereby uses different method matchings than adaptation. Future work should implement the proposed ideas in this paper around our adapter generation tool to provide a working application. The overcoming of signature mismatches is a big challenge, therefore, more research is needed in this area to further push fully automated adaptation.

References

[1] S. Becker, A. Brogi, I. Gorton, S. Overhage, A. Romanovsky, and M. Tivoli, "Towards an engineering approach to component adaptation," in Architecting Systems with Trustworthy Components, vol. 3938, 2009, pp. 193–215, ISSN: 0302-9743.

[2] B. H. Liskov and J. M. Wing, "A behavioral notion of subtyping," in ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 16, no. 6, 1994, pp. 1811–1841.

[3] D. Yellin and R. Strom, "Protocol specifications and component adaptors," in ACM Transactions on Programming Languages and Systems (TOPLAS), vol. 19, no. 2, 1997, pp. 292–333.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education, 1994, ISBN: 9780321700698.

[5] O. Hummel and C. Atkinson, "The managed adapter pattern: Facilitating glue code generation for component reuse," in Formal Foundations of Reuse and Domain Engineering, S. Edwards and G. Kulczycki, Eds. Springer Berlin Heidelberg, 2009, pp. 211–224.

[6] M. Fowler, Patterns of Enterprise Application Architecture. Addison-Wesley, 2003, ISBN: 978-0321127426.

[7] O. Hummel and C. Atkinson, "Automated creation and assessment of component adapters with test cases," in Component-Based Software Engineering, L. Grunske, R. Reussner, and F. Plasil, Eds. Springer Berlin Heidelberg, 2010, pp. 166–181.

[8] "The Adapter Generation Tool," 2015, URL: http://oliverhummel.com/adaptation/tool.zip [accessed: 2015-01-02].

[9] D. Seiffert and O. Hummel, "Improving the runtime-processing for component adaptation," in Lecture Notes in Computer Science, Springer, J. Favaro and M. Morisio, Eds. Springer Berlin Heidelberg, 2013, pp. 81–96.

[10] D. Seiffert and O. Hummel, "Adapting arrays and collections: Another step towards the automated adaptation of object ensembles," in Lecture Notes in Computer Science, Springer, I. Schaefer and I. Stamelos, Eds., vol. 8919. Springer International Publishing Switzerland, 2015, pp. 348 – 363.

[11] T. Gschwind, Type Based Adaptation: An Adaptation Approach for Dynamic Distributed Systems, A. Coen-Porisini and A. van der Hoek, Eds. Springer Berlin Heidelberg, 2003.

 ISBN: 978-1-61208-391-9

[12] S. Nasehi and F. Maurer, “Unit tests as api usage examples,” in International Conference on Software Maintenance (ICSM). IEEE, 2010, pp. 1–10, ISSN: 1063-6773.

[13] S. Galler and B. Aichernig, “Survey on test data generation tools,” in International Journal on Software Tools for Technology Transfer, vol. 16, no. 6, pp. 727–751, 2013, ISSN: 1433-2787.

[14] X. Pintando and B. Junod, “Gluons: Support for software component cooperation,” in Object Frameworks, D. Tsichritzis, Ed. Universite De Geneve, Switzerland, 1992.

[15] L. Cavallaro, E. Di Nitto, P. Pelliccione, M. Pradella, and M. Tivoli, “Synthesizing adapters for conversational web-services from their wsdl interface,” in Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2010, pp. 104–113.

[16] H. R. Motahari Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, “Semi-automated adaptation of service interactions,” in Proceedings of the 16th international conference on World Wide Web, 2007, pp. 993–1002.

[17] D. Brenner, C. Atkinson, B. Paech, R. Malaka, M. Merdes, and D. Suliman, “Redicing verification effort in component-based software engineering through built-in testing,” in Information Systems Frontiers, vol. 9, no. 2-3, 2007, pp. 151–162.

[18] G. A. Cohen, J. S. Chase, and D. L. Kaminsky, “Automatic program transformation with joie,” in Proceedings of the USINEX 1998 Annual Technical Conference, 1998, pp. 167 – 178.

[19] S. P. Reiss, “Semantics-based code search,” in Proceedings of the 31st International Conference on Software Engineering (ICSE 2009). IEEE Computer Society, 2009, pp. 243 – 253, ISBN: 978-1-4244-3453-4.

[20] U. Hoelzle, “Integrating independently-developed components in object-oriented languages,” in ECOOP 93Object-Oriented Programming, 1993, pp. 36–56.

[21] S. Kell, “Component adaptation and assembly using interface relations,” in OOPSLA ’10 Proceedings of the ACM international conference on Object oriented programming systems languages and application, vol. 45, no. 10, 2010, pp. 322–340.

[22] M. Eslamichalandar, K. Barkaoui, H. Reza, and H. Motahari-Nezhad, “Service composition adaptation: an overview,” in Second International Workshop on Advanced Information Systems for Enterprises (IWAISE), 2012, pp. 20–27.

 ISBN: 978-1-61208-391-9

An Emerging Automation Framework for Adaptive Video Games

Muhammad Iftekher Chowdhury, Michael Katchabaw
Department of Computer Science
University of Western Ontario
London, Canada
{iftekher.chowdhury, katchab}@uwo.ca

Abstract—**Previous attempts at adaptive video games can be characterized as ad-hoc from a software engineering perspective; lacking rigor, structure, and reusability, with custom solutions per game. There is a critical need for software frameworks, patterns, libraries, and tools to enable adaptive systems for games. In this paper, we present architecture of a semi-automatic framework that leverages code generation based on design patterns to introduce adaptability in video games. We also discuss key responsibility and implementation choices for each components of the framework.**

Keywords-adaptive video game; design patterns; game development process; software quality; adaptibility automation

I. INTRODUCTION

It is becoming increasingly clear that games must be adaptive in nature — malleable and able to reshape to the needs, expectations, and preferences of the player [2]. Adaptive systems are designed to excel at situations that cannot be completely or singularly modeled prior to development, and so they must be able to satisfy requirements that arise only after they are put in use; this is very much the case in games. Nearly every aspect of a game can be made adaptive in this way: the game world (structural elements, composition); the population of the world (the agents or characters in the world); any narrative elements (story, history, or back- story); gameplay (challenges, obstacles); the presentation of the game to the player (visuals, music, sound); and so on. In being adaptive, games can provide more compelling, engaging, immersive, and perhaps personalized or customized experiences to their player, leading to a significantly better outcome for the player, and far more success for the game in the end [2].

The software engineering literature on adaptive systems provides various solutions focusing on software requirements, system architectures, software design patterns, and so on. Unfortunately, it is difficult to directly apply adaptive systems work from other domains to video games [10]. Games do more than deliver functionality as in other software systems; there is a larger emphasis on engagement, immersion, and experience, as well as greater demands on interactivity and real-time performance and presence. These factors require careful consideration often not required in other domains.

Furthermore, the adaptive video game literature primarily focuses on algorithms, frameworks, empirical studies and game design activities but rarely takes any benefits from the progress in adaptive system literature. Previous attempts at adaptability in video games can be characterized as ad-hoc from a software engineering perspective; lacking rigor, structure, and reusability, with custom solutions per game, which is not acceptable [10]. There is a critical need for reusable software infrastructure to enable the construction of adaptive games [11]. Addressing this problem is the broad goal of our research. While this is a difficult goal to achieve [2], both from theoretical and practical perspectives, we have found success in this area by leveraging software design patterns [1].

In our earlier work [11], we discussed our design pattern based approach to adaptive games and demonstrate the effectiveness of our approach through case studies. Our current goal is to create tool support that will assist developers in introducing adaptability in video games using the design patterns. Existing literature (e.g., [14][20]) suggest that design patterns are specifically suitable for code generation. We also noticed a high percentage of code reusability while using these design patterns during our earlier case studies. Motivated from these points, in this paper, we present the architecture of a semi-automatic framework that leverage code generation based on design patterns to introduce adaptability in video games.

The rest of this paper is organized as follows. In Section II, we discuss the literature reviewed. In Sections III and IV, we describe the design patterns for adaptive video games and motivation behind our current work. In Section V, we present architecture of a semi-automatic framework that leverage code generation based on design patterns to introduce adaptability in video games. In Section VI, we conclude the paper.

II. RELATED WORK

In recent years, adaptive video games and auto dynamic difficulty have received notable attention from numerous researchers. Some of this research is primarily focused on knowledge seeking, whereas other works present solutions such as frameworks and algorithms. Additionally, in some research, new solutions are presented together with empirical validations. In below sub-sections, we review some of these works.

A. Adaptive Game

In the highly influential work [4], Charles and Black propose a framework for adaptive video games incorporating ideas of player-centered game design comprising four key aspects: player modeling, adaptive game environments in

 ISBN: 978-1-61208-391-9

response to player needs, monitoring the effectiveness of any adaptation, and dynamic player classification. They also proposed several neural network approaches for instantiating this framework.

Andrade et al. [7] developed a 2D fighting game where players utilized one of the four strategies: random, state-based, traditional (optimal) reinforcement learning (ORL agent), and adaptive reinforcement learning (ARL agent). Their results showed that the ARL agent was able to adapt to all three types of opponents with a relatively small number of games played.

Togelius et al. [9] attempted to evolve tracks of racing games that fit the players' driving styles to increase overall entertainment. Tracks were given a number of control points based on different implementations and the adaptation algorithm used these control points as locations to alter the shape of the track. They found that using a segment based method of control point distribution resulted in tracks having long straight paths for beginner players and sharper turns for advanced players.

B. Auto Dynamic Difficulty

Bailey and Katchabaw [3] developed an experimental testbed based on Epic's Unreal engine that can be used to implement and study auto dynamic diificulty in games. A number of mini-game gameplay scenarios were developed in the test-bed and these were used in preliminary validation experiments.

Rani et al. [17] suggested a method to use real time feedback, by measuring the anxiety level of the player using wearable biofeedback sensors, to modify game difficulty. They conducted an experiment on a Pong-like game to show that physiological feedback based difficulty levels were more effective than performance feedback to provide an appropriate level of challenge. Physiological signals data were collected from 15 participants each spending 6 hours in cognitive tasks (i.e., anagram and Pong tasks) and these were analyzed offline to train the system.

Hunicke [18] used a probabilistic model to design adaptability in an experimental first person shooter (FPS) game based on the Half-life SDK. They used the game in an experiment on 20 subjects and found that adaptive adjustment increased the player's performance (i.e., the mean number of deaths decreased from 6.4 to 4 in the first 15 minutes of play) and the players did not notice the adjustments.

Hao et al. [19] proposed a Monte-Carlo Tree Search (MCTS) based algorithm for auto dynamic difficulty to generate intelligence of non player characters. Because of the computational intensiveness of the approach, they also provided an alternative based on artificial neural networks (ANN) created from the MCTS. They also tested the feasibility of their approach using Pac-Man.

Hocine and Gouaïch [16] described an adaptive approach for pointing tasks in therapeutic games. They introduced a motivation model based on job satisfaction and activation theory to adapt the task difficulty. They also conducted preliminary validation through a control experiment on eight healthy participants using a Wii balance board game.

III. DESIGN PATTERNS

In this section, we briefly discuss the four design patterns for enabling adaptability in video games. For further details, the reader is encouraged to refer to [10] for elaborated discussion and examples.

A. Sensor Factory

The sensor factory pattern is used to provide a systematic way of collecting data on a game and its players, and provide those data to the rest of the adaptive system. *Sensor* (please see Figure 1) is an abstract class that encapsulates the periodical collection and notification mechanism. A concrete sensor realizes the Sensor and defines specific data collection and calculation. The *SensorFactory* class uses the "factory method" pattern to provide a unified way of creating any sensors. Before creating a sensor, the *SensorFactory* checks in the *Registry* data structure to see whether the sensor has already been created. If created, the *SensorFactory* just returns that sensor instead of creating a new one. Otherwise, it verifies with a *ResourceManager* whether a new sensor can be created without violating any resource constraints.

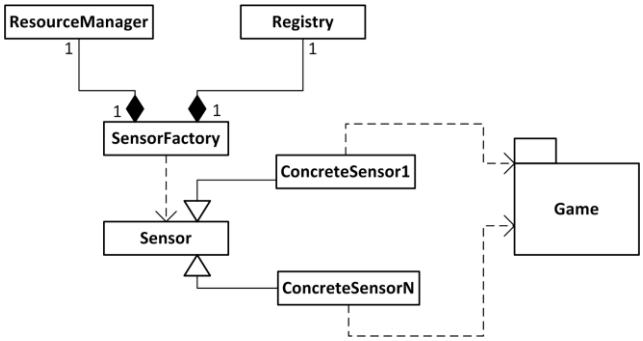


Figure 1. *Sensor factory* design pattern.

B. Adaptation Detector

With the help of the sensor factory pattern, the *AdaptationDetector* (please see Figure 2) deploys a number of sensors in the game and attaches observers to each sensor.

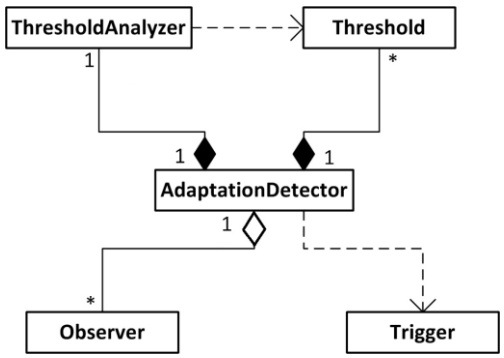


Figure 2. *Adaptation detector* design pattern

Observer encapsulates the data collected from sensor, the unit of data (i.e., the degree of precision necessary for each particular type of sensor data), and whether the data is up-to-

date or not. *AdaptationDetector* periodically compares the updated values found from *Observer*s with specific *Threshold* values with the help of the *ThresholdAnalyzer*. Each *Threshold* contains one or more boundary values as well as the type of the boundary (e.g., less than, greater than, not equal to, etc.). Once the *ThresholdAnalyzer* indicates a situation when adaptation might be needed, the *AdaptationDetector* creates a *Trigger* with the information that the rest of the adaptation process might need.

C. Case Based Reasoning

While the adaptation detector determines the situation when an adjustment is required by creating a *Trigger*, case based reasoning (please see Figure 3) formulates the *Decision* that contains the adjustment plan. The *InferenceEngine* has two data structures: the *TriggerPool* and the *FixedRules*. *FixedRules* contains a number of *Rule*s. Each *Rule* is a combination of a *Trigger* and a *Decision*. The *Trigger*s created by the adaptation detector will be stored in the *TriggerPool*. To address the triggers in the sequence they were raised in, the *TriggerPool* should be a FIFO data structure. The *FixedRules* data structure should support search functionality so that when the *InferenceEngine* takes a *Trigger* from the *TriggerPool*, it can scan through the *Rule*s held by *FixedRules* and find a *Decision* that appropriately responds to the *Trigger*.

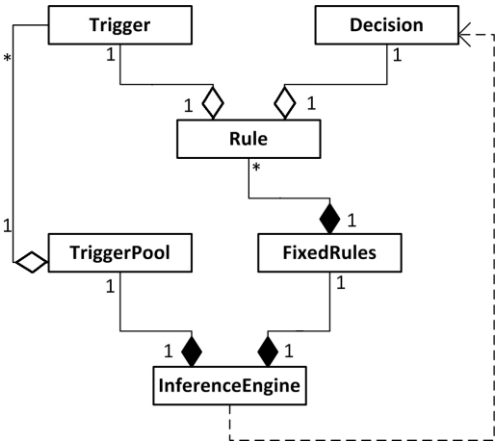


Figure 3. *Case based reasoning* design pattern

D. Game Reconfiguration

Once the adaptive system detects that an adjustment is necessary, and decides what and how to adjust the various game components, it is the task of the game reconfiguration pattern (please see Figure 4) to facilitate smooth execution of the decision. The *AdaptationDriver* receives a *Decision* selected by the *InferenceEngine* (please see case based reasoning in previous subsection) and executes it with the help of the *Driver*. *Driver* implements the algorithm to make any attribute change in an object that implements the *State* interface (i.e., that the object can be in ACTIVE, BEING_ACTIVE, BEING_INACTIVE or INACTIVE states, and outside objects can request state changes). As the name suggests, in the active state, the object shows its usual behavior whereas in the inactive state, the object stops its regular tasks and is open to changes. In the being inactive state, the game finishes the existing tasks based on the already processed player inputs but does not start any new task. In the being active state, the game does not start task based on player input and is not open to any new changes. The *Driver* takes the object to be reconfigured, details of the attribute to be changed and the changed attribute value as inputs. The *Driver* requests the object that needs to be reconfigured to be inactive. When the object becomes inactive, it reconfigures the object as specified. After that, it requests the object to be active and informs the *AdaptationDriver* when the object becomes active. The *GameState* maintains a *RequestBuffer* data structure to temporarily store the inputs received during the inactive state of the game. The *GameState* overrides Game's event handling methods and game loop to implement the *State* interface.

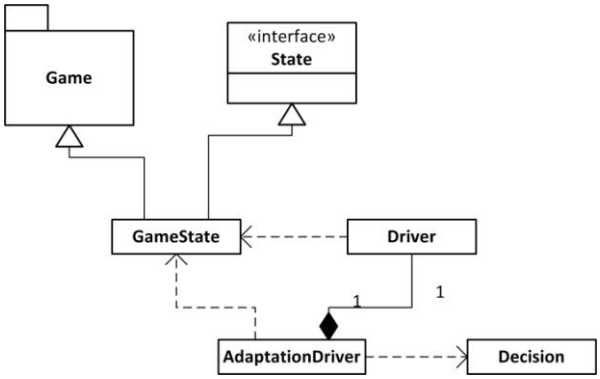


Figure 4. *Game reconfiguration* design pattern

E. Integration of Design Patterns

In [15], Salehie and Tahvildari described integration of four generic steps for an adaptation process namely monitoring, detecting, deciding, and acting. The four design patterns discussed in previous sub-sections work on the same process flow. In this Section, we briefly re-discuss how they work together to create a complete adaptive system (please see Figure 5).

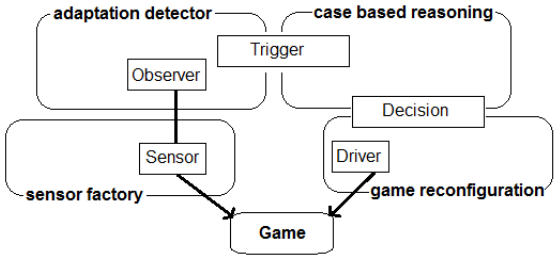


Figure 5. Four design patterns working together in a game

The sensor factory pattern uses Sensors to collect data from the game so that the player's perceived level of difficulty can be measured. The adaptation detector pattern observes Sensor data using Observers. When the adaptation

detector finds situations where difficulty needs to be adjusted, it creates Triggers with appropriate additional information. Case based reasoning gets notified about required adjustments by means of Triggers. It finds appropriate Decisions associated with the Triggers and passes them to the adaptation driver. The adaptation driver applies the changes specified by each Decision to the game, to adjust the difficulty of the game appropriately, with the help of the Driver. The adaptation driver also makes sure that the change process is transparent to the player. In this way, all four design patterns work together to create a complete adaptive system for a particular game.

F. Achieving Adaptive Gameplay

So far we have used these design patterns for implementation of a specific type of adaptability in video games known as auto dynamic difficulty. But in principle these design patterns should be sufficient to implement more complex form of adaptability in game-play.

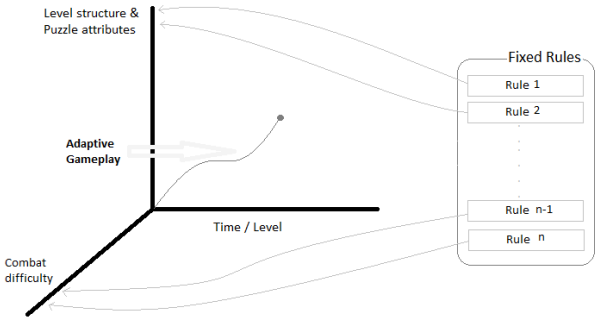


Figure 6. Concept of multi dimensional adaptive gameplay

Figure 6 depicts our position of a multidimensional adaptive game-play. For example, we have chosen two aspects of the game to adjust adaptively. One is *level structure and puzzle attributes*. And the other is *combat difficulty*. There are number of rules and other associated artefacts (i.e., sensors, observers, triggers and decisions) focused on each of these aspects. In a particular *level structure and puzzle attributes* with minimum *combat difficulty* the player may experience a maze type game whereas with a high *combat difficulty* and simple *level structure and puzzle attributes* the player may experience a fighting game. Nearly every aspect of a game can be made adaptive in this way: the game world (structural elements, composition); the population of the world (the agents or characters in the world); any narrative elements (story, history, or back-story); game-play (challenges, obstacles); the presentation of the game to the player (visuals, music, sound); and so on.

IV. Motivation for Automation

In this section, we discuss two key motivations behind our automation effort: the repeatable nature of the process for applying the design patterns and source code reusability achieved through the usage of the design patterns for implementing adaptability in video games.

A. Repeatable Process

Applying our software design pattern based framework for adaptability to a large commercial-scale game such as Minecraft [13], seemed to be a daunting task, at least on the surface. Thus, the process described in Table I was developed to formalize our experiences from using it in Pac-Man [10] and TileGame [11] to assist in the adaptability-enablement of larger games such as Minecraft. In practice, we found that applying such a methodical process enabled adaptability in Minecraft quite readily, and that our approach was easily adapted for use in this rather foreign environment with no more significant changes than we found in our earlier work with much simpler games. This is a key motivation for our current work as concrete activities (such as the ones in Table I) are easier to build a tool upon.

TABLE I. Adaptive Game Implementation Process

#	Activity	Output
1	Identify the aspects of the game that will be adaptively adjusted.	
2	For each of the aspects identified in step-1 repeat step-3 to step-9.	
3	Define or reuse available sensors.	Sensors
4	Identify or introduce attributes that can be adjusted.	
5	Identify adaptation scenarios involving sensors and attributes from step-3 and step-4.	
6	Define thresholds based on the scenarios identified in step-5 for the sensors defined in step-3, and define observers to relate thresholds to sensors.	Thresholds, Observers
7	Define triggers to represent each scenario, and develop the adaptation detector logic based on the scenarios.	Triggers
8	Use attributes identified in step-4 to create decisions to modify game difficulty according to the scenarios identified in step-5.	Decisions
9	Define rules to relate triggers to decisions based on the adaptation scenarios identified in step-5.	Rules

B. Reusable Source Code

We have also carried out a source code analysis of these games. In [11], the Minecraft adaptability implementation was compared to the adaptability implementations of Pac-Man and TileGame. During this analysis, we have noticed that a large percentage of the resultant code is generalization and instantiation of other high level classes (e.g., Sensors, Triggers, Thresholds, and Decisions etc.).

TABLE II. Categorization of the Adaptive Source Code

Category of Source Code	SLOC	%
Completely reusable	600	74.26
Specialization (Concrete Sensors (64) and Concrete Decisions (22))	86	10.64
Instantiation (Adaptation Detector (70) and Inference Engine (11))	81	10.02
Other logic	41	5.07
Total	808	

 ISBN: 978-1-61208-391-9

In Table II, we provide a summary of the analysis derived from the results presented in [11]. Here, we can see that 74.26% source code remained the same from earlier projects. Also, 10.64% source code is specializations and 10.02% code is for instantiation. Only 5.07% source code is other specific game logic. The specialization and instantiation (20.66%) related source code of the adaptive system consists of similar looking classes and statements. This result motivates us to create a tool that will allow us to develop and maintain these artifacts in a semi-automatic manner.

V. AUTOMATION FRAMEWORK

Figure 7 depicts a high level decomposition of our semi-automatic system. The key idea is to represent part of the adaptive logic as a relational model that is mutable. The core software elements are divided into four components: (i) Collector and Executor, (ii) Enhancer, (iii) Manager, and (iv) Translator. The collector and executor component interfaces the relational model with the game in question. It collects meta-information from the game's source code as well as runtime logging information and passes that to the model. It can also execute modification instructions presented in the model. The manager component provides graphical user interfaces to easily manipulate the model. The enhancer component facilitates the decision making process (i.e., when, how and to what degree to modify the game). The purpose of the translator component is translating the relational model, when finalized, to executable software artifacts (i.e., source code). In the following subsections, we discuss each of these components in further detail.

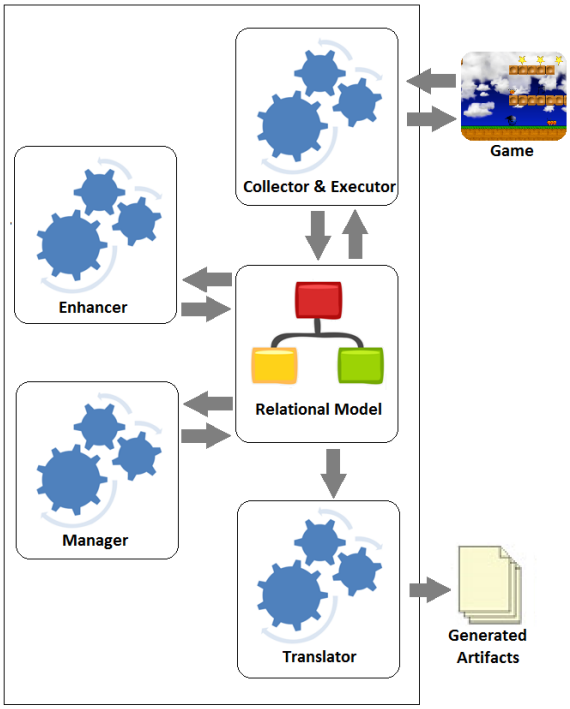


Figure 7. Components of the semi-automatic framework

A. Relational Model

Central to the framework is a relational model, as all the other components use it as a repository for all of their information. This is essentially storage for a set of objects and relations that represent much of the dynamic information (e.g., Sensor's name, relations between sensors and attributes, etc.) for an intended adaptive system as well as some meta-information (e.g., attributes, logging information, etc.). The structure of the model is derived from the design patterns described earlier and is not dependent on the platform or genre of the video game. There should be appropriate APIs for other components to collect information from the model. Implementation choices for the relational model include databases, XML storage, file based data structures, amongst others.

B. Collector and Executor

The collector and executor component interfaces the relational model with the game and thus should depend on the platform of the game. The collector needs to be configured with some base level objects (e.g., game world, player, enemies, inventory etc.). For the rest of the system to work, the collector needs to conduct a Breadth-First Search (BFS) starting from those base level objects and populate the model with a list of attributes and related data types using a hierarchical storage method such as recursive relations. Many languages provide programmatic ways (e.g., Java reflection) to collect such information with ease.

We have identified some key challenges regarding the implementation of the executor and the relational model:

- Identifying the depth of the object hierarchy to search,

- Representing relationships other than hierarchical ones and representing shared objects,

- Representing any run time changes on the hierarchy.

The executor can execute modification instructions presented as decisions in the relational model and the collector can collect more information based on those modifications.

C. Manager

The manager is another generic component that does not need to be aware of the details of the rest of the system and the platform other than the relational model. It is a collection of graphical user interfaces and business logic to easily manage the relational model. Once the attributes are recorded by the collector, they can be marked to be monitored using this component.

D. Enhancer

The enhancer is also a generic component and only needs to interact with the model and thus can be implemented in any language and need not be aware of the game's platform. It is a collection of tools that helps the game designer or developer to make decisions about which attributes to monitor, threshold values, which attributes to modify and to what degree, amongst others. It usually works on data

 ISBN: 978-1-61208-391-9

collected by the collector. Here we give examples of such tools:

- Statistical analysis: Such as factor and co-relation analysis.

- Graphical analysis: Such as curve fitting.

- Machine learning: For example, in [6], Southey et al. described an active learning based semi-automatic gameplay analysis tool that interacts with game-engine or frameworks like this one through an abstraction layer and mainly consists of a sampler, a learner and a visualizer component. The usage of the tool is demonstrated in commercial context (i.e., Electronic Art's [5] FIFA'99).

E. Translator

The translator component needs to be aware of the platform of the video game and needs to generate the artifacts accordingly. It can either directly translate to source code or generate an intermediate marked up description suitable for other code generation tools. The code generation logic is often quite straight-forward. For each file (e.g., Java class), the static parts of the code need to be predefined and the translator injects the dynamic parts as necessary. Please see literature on source code generation (e.g., [8]) for elaborated discussion and methodologies.

Benefits of such semi automatic tools include reducing efforts and defects, standardization, ease of progress measurability and improving maintainability, etc.

VI. CONCLUDING REMARKS

There is a critical need for software frameworks, patterns, libraries, and tools to enable adaptive systems for games. We have found success in this area by leveraging software design patterns. In this paper, we present architecture of a semi-automatic framework that leverage code generation based on design patterns to introduce adaptability in video games. Benefits of such a tool include minimizing developer efforts and increasing maintainability. We designed the framework following a loosely coupled architecture that is generalizable across various platforms. We will discuss a prototype (preliminary discussion of a proof-of-concept prototype and its usage can be found in [12]) based on this framework in our subsequent work. We also encourage other researchers to extend our framework as appropriate.

REFERENCES

[1] A. J. Ramirez and B. H. Cheng, "Design patterns for developing dynamically adaptive systems," In Proceedings of 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2010, pp. 49 - 58.

[2] A. Glassner, Interactive Storytelling: Techniques for 21st Century Fiction. A K Peters, Ltd., 2004.

[3] C. Bailey and M. Katchabaw, "An experimental test bed to enable auto-dynamic difficulty in modern video games," In Proceedings of the 2005 North American Game-On Conference, 2005, pp. 18-22.

[4] D. Charles and M. Black, "Dynamic Player Modelling: A Framework for Player-Centered Digital Games," In Proceedings of the International Conference on Computer Games: Articial Intelligence, Design and Education, 2004, pp. 8-10.

[5] EA Sports, Retrieved from: http://www.easports.com/. Last accessed: Feb 14, 2015.

[6] F. Southey, G. Xiao, R. C. Holte, M. Trommelen, and J. Buchan, "Semi-Automated Gameplay Analysis by Machine Learning," in Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-05), 2005, pp. 123-128.

[7] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Challenge-sensitive action selection: an application to game balancing," In Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2005, pp. 194-200.

[8] Herrington, J. "Code Generation in Action,". Manning Publications Co., 2003.

[9] J. Togelius, R. D. Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," In Proceedings of the IEEE International Symposium on Computation Intelligence and Games (CIG 2007), 2007, pp. 252-259.

[10] M. I. Chowdhury and M. Katchabaw, "Software design patterns for enabling auto dynamic difficulty in video games," In 17 International Conference on Computer Games, 2012, pp. 76 - 80.

[11] M. I. Chowdhury and M. Katchabaw, " A Software Design Pattern Based Approach to Adaptive Video Games," In Proceedings of the 5th International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE-2013), 2013, pp. 40 - 47.

[12] M. I. Chowdhury. 2014 "A Software Design Pattern Based Approach to Auto Dynamic Difficulty in Video Games," PhD Thesis, University Of Western Ontario. Can be accessed online at: http://ir.lib.uwo.ca/etd/2522/. Last accessed: Feb 14, 2015.

[13] Minecraft, Retrieved from: https://minecraft.net/. Last accessed: Feb 14, 2015.

[14] M. Ohtsuki, N. Yoshida, and A. Makinouchi, "A source code generation support system using design pattern documents based on SGML," in Proceedings of the 6th Asia Pacific Software Engineering Conference (APSEC '99) , pp.292-299.

[15] M. Salehie and L. Tahvildari, " Self-Adaptive Software: Landscape and Research Challenges," In ACM Transactions on Autonomous and Adaptive Systems, 2009, May 2009, Vol. 4, No. 2, Article 14.

[16] N. Hocine and A. Gouaïch, " Therapeutic games' difficulty adaptation: An approach based on player's ability and motivation," In Proceedings of 16th International Conference on Computer Games (CGAMES), 2011, pp. 257 - 261.

[17] P. Rani, N. Sarkar, and C. Liu, "Maintaining optimal challenge in computer games through real-time physiological feedback," In Proceedings of 11th International Conference on Human-Computer Interaction, 2005, pp. 184-192.

[18] R. Hunicke, "The case for dynamic difficulty adjustment in games," In Proceedings of 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (ACE 2005), 2005, pp. 429-433.

[19] Y. Hao, S. He, J. Wang, X. Liu, J. Yang, and W. Huang, "Dynamic difficulty adjustment of game AI by MCTS for the game Pac-Man," In Proceedings of 6th International Conference on Natural Computation (ICNC 2010), 2010, pp. 3918-3922.

[20] Y. Seo and Y. Song. 2006. A study on automatic code generation tool from design patterns based on the XMI. In 2006 International Conference on Computational Science and Its Applications - Volume Part IV, pp. 864-887.

 ISBN: 978-1-61208-391-9