ADAPTIVE 2011

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

ISBN: 978-1-61208-156-4

September 25-30, 2011
Rome, Italy

ADAPTIVE 2011 Editors

Jorge Fox, Institute of Information Science and Technology, CNR, Italy

Andreas Rausch, Technische Universität Clausthal, Germany

ADAPTIVE 2011

Foreword

The Third International Conference on Adaptive and Self-Adaptive Systems and Applications [ADAPTIVE 2011], held between September 25 and 30, 2011 in Rome, Italy, targeted advanced system and application design paradigms driven by adaptiveness and self-adaptiveness. With the current tendencies in developing and deploying complex systems, and under the continuous changes of system and application requirements, adaptation is a key feature. Speed and scalability of changes require self-adaptation for special cases. How to build systems to be easily adaptive and self-adaptive, what constraints and what mechanisms must be used, and how to evaluate a stable state in such systems are challenging duties. Context-aware and user-aware are major situations where environment and user feedback is considered for further adaptation.

We take here the opportunity to warmly thank all the members of the ADAPTIVE 2011 Technical Program Committee, as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and efforts to contribute to ADAPTIVE 2011. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations, and sponsors. We are grateful to the members of the ADAPTIVE 2011 organizing committee for their help in handling the logistics and for their work to make this professional meeting a success.

We hope that ADAPTIVE 2011 was a successful international forum for the exchange of ideas and results between academia and industry and for the promotion of progress in the field of adaptive and self-adaptive systems and applications.

We are convinced that the participants found the event useful and communications very open. We also hope the attendees enjoyed the charm of Rome, Italy.

ADAPTIVE 2011 Chairs:

Antonio Bucchiarone, FBK-IRST of Trento, Italy
Radu Calinescu, Aston University, UK
Weirong Jiang, Juniper Networks Inc. - Sunnyvale, USA
Serge Kernbach, University of Stuttgart, Germany
Thomas H. Morris, Mississippi State University, USA
Dalimír Orfánus, ABB Corporate Research Center, Norway

ADAPTIVE 2011

Committee

ADAPTIVE Advisory Chairs

Radu Calinescu, Aston University, UK
Thomas H. Morris, Mississippi State University, USA
Serge Kernbach, University of Stuttgart, Germany
Antonio Bucchiarone, FBK-IRST of Trento, Italy

ADAPTIVE 2011 Industry/Research Chairs

Dalimír Orfánus, ABB Corporate Research Center, Norway
Weirong Jiang, Juniper Networks Inc. - Sunnyvale, USA

ADAPTIVE 2011 Technical Program Committee

Sherif Abdelwahed, Mississippi State University, USA
Habtamu Abie, Norwegian Computing Center - Oslo, Norway
Mahmood Ahmadi, TU Delpt, The Netherlands
José M. Alcaraz Calero, Hewlett-Packard Laboratories - Bristol, UK
Mohammad Ali, Jinnah University, Islamabad, Pakistan
Giner Alor Hernández, Instituto Tecnológico de Orizaba - Veracruz, México
Djamel Belaid, IT SudParis, France
Imen Ben Lahmar, IT-SudParis, France
Antonio Bucchiarone, FBK-IRST of Trento, Italy
Radu Calinescu, Aston University, UK
Aldo Campi, University of Bologna - Cesena, Italy
Valerie Camps, Universite Paul Sabatier – Toulouse, France
Bogdan Alexandru Caprarescu, West University of Timisoara, Romania
Carlos Carrascosa, Universidad Politécnica de Valencia, Spain
Eduardo Cerqueira, Federal Unversity of Para - Brazil
Keith C.C. Chan, The Hong Kong Polytechnic University - Hung Hom, Kowloon, Honk Kong
Sheng-Wei (Kuan-Ta) Chen, Academia Sinica, Taiwan
Po-Hsun Cheng, National Kaohsiung Normal University, Taiwan
José Alfredo F. Costa, Federal University, UFRN, Brazil
Carlos Cuesta, Rey Juan Carlos University, Madrid, Spain
Michel Diaz, LAAS - Toulouse, France
Mihaela Dinsoreanu, Technical University of Cluj-Napoca, Romania & University of Maryland University College, USA
Ioanna Dionysiou, University of Nicosia, Cyprus
Reuben A. Farrugia, University of Malta, Malta
Alois Ferscha, Johannes Kepler Universität Linz, Austria
Ziny Flikop, Consultant, USA
Adina Magda Florea, University "Politehnica" of Bucharest, Romania
Carlos Flores, Universidad de Colima, México

Naoki Fukuta, Shizuoka University, Japan
Francisco José García Peñalvo, Universidad de Salamanca, Spain
Susan Gauch, University of Arkansas, USA
George Giannakopoulos, University of Trento, Italy
Georgios K. Giannikis, University of Thessaly - Volos, Greece
Harald Gjermundrod, University of Nicosia, Cyprus
Gregor Grambow, Aalen University, Germany
Cathal Gurrin, Dublin City University, Ireland
Mirsad Hadzikadic, UNC Charlotte, USA
Leszek Holenderski, Philips Research - Eindhoven, The Netherlands
Marc-Philippe Huget, Polytech Annecy-Chambery-LISTIC/University of Savoie, Annecy Le Vieux, France
Jean-Paul Jamont, Université Pierre Mendès, France - IUT de Valence, France
Jianmin Jiang, University of Bradford, UK
Weirong Jiang, Juniper Networks Inc. - Sunnyvale, USA
Ilia Kabak, "STANKIN" Moscow State Technological University, Russia
Elsy Kaddoum, Paul Sabatier Toulouse, France
Anthony Karageorgos, Technological Educational Institute of Larissa/Karditsa Branch, Greece
Hamid Reza Karimi, University of Agder - Grimstad, Norway
John Keeney, Trinity College Dublin. Ireland
Serge Kernbach, University of Stuttgart, Germany
Franziska Klugl, Orebro University, Sweden
Satoshi Kurihara, Osaka University, Japan
Rico Kusber, University of Kassel, Germany
Mikel Larrea, University of the Basque Country, UPV/EHU, Spain
Jingpeng Li, The University of Nottingham, UK
Henrique Lopes Cardoso, University of Porto, Portugal
Emiliano Lorini, Institut de Recherche en Informatique de Toulouse (IRIT), France
René Mandiau, Université de Valenciennes et du Hainaut-Cambrésis, France
Massimo Marchiori, University of Padova, Italy
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal
Olga Melekhova, Université Pierre & Marie Curie - LIP6/CNRS, France
Frédéric Migeon, IRIT - Toulouse, France
Tommy Morris, Mississippi State University, USA
Gero Mühl, Universität Rostock, Germany
Filippo Neri, Università di Napoli "Federico II", Italy
Flavio Oquendo, European University of Brittany - UBS/VALORIA, France
Dalimír Orfánus, ABB Corporate Research Center, Norway
Stefan Poslad, Queen Mary University of London, UK
Claudia Raibulet, Universitàdegli Studi di Milano-Bicocca, Italy
Mahesh Raisinghani, TWU School of Management - Denton, USA
Sitalakshmi Ramakrishnan, Monash University, Australia
Sebastian Ries, CASED - Center for Advanced Security Research Darmstadt, Germany
Alejandro Rodríguez González, Universidad Carlos III de Madrid, Spain
Yacine Sam, Université François Rabelais Tours, France
Ava Fatah gen. Schieck, Bartlett UCL - London UK
Michael Schumacher, University of Applied Sciences Western Switzerland, Switzerland
Sebastian Senge, Technical University of Dortmund, Germany
Vasco Soares, Instituto de Telecomunicações / University of Beira Interior / Polytechnic Institute of

Castelo Branco, Portugal
Christoph Sondermann-Wölke, University of Paderborn, Germany
Sofia Stamou, University of Patras & Ionian University, Greece
Vladimir Stantchev, Berlin Institute of Technology, Germany
Yéhia Taher, The European Research Institute in Service Science / Tilburg University, The Netherlands
Sotirios Terzis, University of Strathclyde - Glasgow, UK
Catherine Tessier, Onera – Toulouse, France
Michael Zapj, Universität Kassel, Germany

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

Facilitating Context-Awareness in Composite Mashup Applications

Stefan Pietschmann, Carsten Radeck, and Klaus Meißner
Technische Universität Dresden, Germany
{*Stefan.Pietschmann,Carsten.Radeck,Klaus.Meissner*}*@tu-dresden.de*

***Abstract*—Dynamic adaptation and adaptivity of web applications have been subject to research for over a decade. With the shift from document-centered hypermedia to rich web applications, i. e., software as a service solutions, the applicability of traditional adaptation methods and techniques is in question. We first investigate what it means to facilitate adaptation within composite service-oriented applications and deduce adequate adaptation techniques. Then, we introduce a generic adaptation system which can be integrated with existing composition platforms to facility runtime adaptivity on a component and composition level, based on a platform-independent definition of adaptive behavior. With the help of a comprehensive sample application we eventually show the suitability and practicability of our approach.**

***Keywords*-mashups; composite applications; context-awareness; adaptation; CRUISe.**

I. Introduction

The Internet has become an open application platform and is not anymore a static, document-centered source of information. Based on the service-oriented paradigm, *mashups*, i. e., applications composing distributed web resources, have gained momentum. While initially their use was limited to the integration of data and business logic based on proprietary platforms, recently the need for *presentation integration* has been underlined [1]. Several research projects have since addressed this problem resulting in lightweight component and composition models that even allow for a *universal composition* [2], [3] of applications. This entails the uniform description and integration of resources covering different application layers, ranging from services encapsulating data and business logic, to user interface parts.

Alongside this development, the context of web applications has become increasingly complex and diverse. Developers are facing a growing heterogeneity of users, devices, and usage situations. Applications can no longer be developed for specific platforms or environments: they shall be available and usable both in the office with a desktop computer and in the tram using an iPad or the like. Users of different skills should be able to interact with them, taking into account specific user abilities and preferences, and also their current situations.

Context-awareness, especially in the web application domain as addressed by our work, has been subject to research for a long time. Yet, most of this research has been targeted at "traditional" and closed-corpus hypermedia systems [4]. Alternative approaches from service computing have studied adaptive service composition at the data and business logic levels, primarily with regard to QoS measures. However, presentation-oriented, universal compositions pose new challenges regarding the dynamic configuration and composition, which have not been addressed by prevalent concepts [5].

The adaptation of composite applications can be looked at from different angles. On the one hand – since we propose a model-driven development [3] – design time concepts for the authoring, reuse, and maintenance of adaptive behavior are needed and discussed in [6]. On the other hand, runtime adaptation involves both the initial context-aware service selection and the adaptation of a composition. This paper focuses on the latter, i. e., the system architecture and mechanisms necessary to monitor, manage, and use context information to dynamically adapt composite mashups.

Starting from traditional adaptation methods and techniques, we investigate into techniques facilitating adaptation and adaptability within mashups. We then present a generic adaptation system designed to be used in conjunction with existing composition environments. It addresses the shortcomings of available concepts and combines context monitoring and management, the platform-independent modeling of adaptive behavior, its evaluation and finally the realization of dynamic adaptations at the component and composition level at runtime. Thereby, adaptive behavior and adaptation techniques are decoupled from concrete platforms and applications, catering reusability, maintenance and extensibility.

In the next section we define the conceptual foundations for our system, present exemplary adaptation scenarios, and deduce corresponding adaptation techniques. Afterwards, we discuss related efforts to enable context-awareness in application and service compositions. After giving a brief overview of our concept, the following section presents the application-independent definition of adaptive behavior, context management facilities, and the realization of dynamic adaptation techniques including component exchange. We then provide details on the implementation of our system, its integration with the CRUISe composition environment [3], and its validation with the help of a sample application. Finally, we conclude this paper and outline future work.

 ISBN: 978-1-61208-156-4

II. Adaptation in Mashup Applications

To facilitate adaptive behavior in mashups, we first need to clarify the relation between traditional *Adaptive Hypermedia* and modern applications "mashed up" from distributed web resources. To this end, this section provides a brief characterization of the type of applications addressed by our work, presents exemplary adaptation use cases and deduces corresponding techniques.

A. Composite Mashup Applications

Mashups indicate a way to create new applications by combining existing web resources utilizing data and web APIs [5]. While originally restricted to data and application logic, there are ongoing efforts to enable compositions including user interfaces. In contrast to most programmatic mashups, we focus on *universal* composition efforts [3], [2] which imply certain component and composition models. Thus, we call them *composite* mashup applications (CMA) to denote their component-based nature.

While *Adaptive Hypermedia* stipulates closed-corpus systems [4] for data-driven, document-centered applications, CMAs are composed from black-box software components. Those can be integrated, used and adapted only via their interfaces, described by WSDL in the case of web services. Most importantly, these "building blocks" are distributed and strongly decoupled by definition. Despite this, the resulting applications are highly interactive, so components must by tightly integrated. Finally, unlike traditional applications using a hypernet of web pages, CMAs are usually single-page solutions offering different views on an application.

It becomes evident that techniques like *page variants*, *sorting fragments*, or *link annotation* can not be applied to CMAs directly, as they are based on different metaphors. While *pages* can be considered *views* in our vocabulary of concepts, and *fragments* may correlate with *components*, the techniques still imply hypertext documents and are rather located *within* components in our concept space.

Next we present a number of exemplary adaptation use cases, which allow us to deduce higher-level adaptation techniques specific to CMAs.

B. Adaptation Use Cases

To illustrate the set of adaptation possibilities, it is necessary to identify the adaptation triggers, i. e., the context information available. Generally, four context categories can be taken into account [7]: First, the *delivery context* defines the technical environment of the application at runtime, e. g., the capabilities of the device, web browser and network connection. The second category includes the *user*, his identity, roles, characteristics, preferences, etc. Third, the *physical environment* is characterized by information such as the location, noise level, and brightness. The last aspect includes *situation and time*, e. g., being at home or at work, the current weather, or the season. Since CMAs are composed from distributed services, the *quality of service* is an additional trigger which has been intensively studied in the context of web service compositions (cf. [8]).

This wide range of context information implies a variety of adaptation scenarios. A special field that has gained lots of popularity are *location-based services*, such as adaptive route planners. The following use cases give a practical insight into the possibilities and challenges of context-awareness: Imagine, you are on vacation and use the services provided by a travel guide application: it shows locations of nearby restaurants and hotels, or sights on a map. For each of them, you can check further details in the form of videos, images, and text, calculate a route. The suitability and usability of the application in this rather simple scenario largely depends on the reasonable integration and use of context knowledge. Examples including the type of context (in parentheses) are:

- The application including all components use your native language (*user*).
- The map initially centers at the your current position by default (*physical*).
- All components are constantly synchronized with your location: the route planner adapts the route calculation once you move (*physical*).
- The selection of locations matches your interests and preferences: a vegetarian user is offered special restaurants that cater his needs (*user*).
- Route planning considers your mobility (foot/car) and whether you have any disabilities that might impact on transfer times (*user and situation*).
- The layout changes with respect to the available screen real estate, and big images are replaced with textual descriptions on mobile devices (*delivery*).
- You can minimize, remove, and exchange components, e. g., to choose your preferred map provider – a choice that is monitored and stored to select the correct component for you in the future (*user*).
- A low battery level on your mobile device triggers the exchange of resource-intensive components, e. g., of a video player with an image slide show (*delivery*).
- Once a component or service becomes unavailable, it is automatically replaced with an alternative (*QoS*).

C. Adaptation Techniques for CMAs

While the abstract *adaptation methods* as presented in [9] and the like remain valid, the corresponding techniques must be updated and applied to the notion of CMAs. We regard *adaptation techniques* as any alteration of an application composition subject to context changes. Hence, adaptation forms an additional aspect that can be applied to a composition model. The reasons therefore range from *adaptive*, *corrective*, *perfective*, to *extensive adaptations* [10]. Consequently, runtime adaptation for CMAs can be realized with the following techniques:

 ISBN: 978-1-61208-156-4

Component Adaptation

- **Adding Components** to the composition supports perfective and extensive adaptations: New components may provide additional data and information to support a user in his task.
- **Removing Components** which are not used or not necessary anymore facilitates perfective adaptations. It must be ensured that removals don't affect application operability and stability, and that data loss is prevented.
- **Exchanging Components** combines the above techniques for corrective and adaptive reasons. Challenges include the state transfer and concurrency problems during the exchange.
- **Reconfiguring Components** basically changes configuration properties, which can result in internal state changes, the use of different algorithms, data sources, etc. This technique is limited to adaptation scenarios foreseen by component developers.
- **Adapting Component Interfaces** implies a change of a component's external interface for interoperability reasons, e. g., when exchanging one component with another. This technique involves both the adaptation of operations, events or the like using adapters, and the mapping or mediation of data.
- **Migrating Components** between different devices, or between client and server, can be seen as a combination of addition and removal along the lines of component *exchange* (see above).

Composition Adaptation

- **Adapting Communication** between components is a powerful technique to adapt control and data flow to the context.
- **Adapting Layout** of the application means to rearrange frontend components on the canvas. This can also involve changes to dimensions of individual components within the layout.
- **Adapting Screenflow** changes the order of views, i. e., which UI component is visible at a certain application state, and the transition between those views.

As stated in [11], both *parameter adaptation* (reconfiguration) and *architectural adaptation* (addition, removal, exchange) are needed and sufficient to support all scenarios of dynamic adaptation. Thus, with the help of the abovementioned techniques, all adaptation examples sketched out before are supported.

An additional dimension of adaptation includes the dynamic reconfiguration and migration of the runtime environment, e. g., the dynamic redistribution between client and server depending on the server load. In this paper, though, we focus on the adaptation of the component-based application within a fixed composition environment.

III. Related Work

Several concepts for runtime adaptation have been proposed. A typical representative from *Adaptive Hypermedia* is AMACONT [12], which offers an extensible context framework – similar to our solution – and realizes adaptation with the help context-dependent variants being part of its document model. In contrast to our work, the concepts are based on dynamically generated documents, and adaptation takes place only when a new document is requested.

Schmidt et al. [13] introduce an event-based adaptation system for RIAs, which employs an ontology-based user model and Event-Condition-Action adaptation rules, both of which served as inspiration for our approach. The solution does not imply the use of any component or composition models, though, and concentrates on event detection and condition evaluation. Consequently, techniques like component exchange are not supported.

ACCADA [14] uses architectural models and an external control loop for context monitoring and dynamic adaptation. Thus, adaptation logic is strictly separated from the application. However, it is used to satisfy functional constraints only – components and adaptation techniques at the presentation layer are not considered. Similarly, the extensive body of research in the field of dynamic service composition fails in this regard. The joint effort here is to enforce composition plans and utility functions. While initially, this involved service exchange with regard to functional and QoS requirements [8], more complex service adaptations [15] and more diverse context information [16] have been used, lately.

Only few academic works have addressed context-awareness for mashup systems, most of which again focus on the initial composition [17] as discussed above. Most importantly, the Mixup framework [18] comes with a component model and an event-based runtime similar to our concepts. *Context components* resemble sensors which publish events upon context changes. A sophisticated context management is not included, though, and the adaptation techniques are quite limited, e. g., adaptive layout and screen flow as well as component exchange are not supported. Further, context components (resembling monitors) are first class concepts of an application, while we strive for decoupling adaptive behavior from application, i. e., composition logic.

In summary, none of the approaches to date provides a generic solution which covers both context management and dynamic adaptation for interactive CMAs. In the next section we introduce our solution to this end.

IV. Adapting Composite Mashups

In this section we present a concept for the runtime adaptation of component-based mashup applications. Therefore, we propose an adaptation system which can be used with existing composition environments – in our case the CRUISe platform [3]. The latter facilitates context-aware component

 ISBN: 978-1-61208-156-4

selection and integration at application initialization time. Based on an abstract composition model, suitable components are dynamically chosen with regard to the particular context. The concept presented here focuses on adaptation at run time, once the application has been initially composed. Following a brief overview of the conceptual architecture, we discuss the definition of adaptive behavior, the strategy to monitor and manage context data, the realization of adaptation techniques, and the adaptation workflow, subsequently.

A. Architectural Overview

As Figure 1 shows, our adaptation infrastructure is divided into two parts responsible for the adaptation and context management, which are coupled with a composition environment.

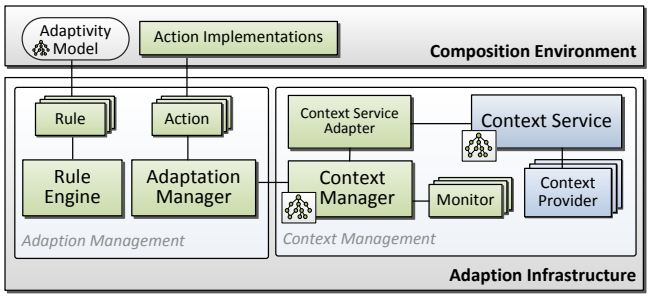


Figure 1. Overview of the adaptation system

Adaptation Management carries out adaptations based on generic Event-Condition-Action (ECA) rules. Those *adaptation rules* address specific parts of a composite application (component properties and instances, layouts, etc.) to be modified by adaptation *actions* subject to (optional) conditions. The evaluation of adaptation rules is carried out by the *Rule Engine*, which subsequently calls the *Adaptation Manager* to execute adaptation actions as specified.

Context Management comprises both context monitoring and modeling. Monitoring within our architecture is carried out by *Monitor* components. Context modeling, reasoning and consistency checking is done by a dedicated system. We argue that this complex task is best externalized, e. g., to a *Context Service* like discussed later. To increase performance and decrease network load, the *Context Manager* holds a local representation of a part of the remote context model and is updated whenever the latter changes. To keep the system flexible and extensible, the *Context Service Adapter* concept hides service specifics and thus makes it easily exchangeable.

The application-specific configuration of the adaptation infrastructure is deduced from platform-specific composition models. They include adaptive behavior, but may also define the supported levels of adaptability and the configuration of context monitors, e. g., update thresholds. Optionally, the context service can provide a structural description of the context model to allow the service adapter to do the necessary mappings.

B. Description of Adaptive Behavior

The definition of adaptive behavior is usually intertwined with application code and heavily depends on the composition platform used. For the adaptation system presented here, we use a platform-independent representation in the form of declarative XML-based ECA rules. They come with certain implications regarding the composition environment, such as event-based communication and a uniform component model. Yet, those are concepts found with the majority of existing mashup systems, ranging from industrial solutions like Yahoo! Pipes to academic approaches such as mashArt [2]. Generally, the rules remain independent from specific composition platforms and languages. They are either written manually, but can also – as in our case – be generated from corresponding information in the composition model [6]. Each of these *Adaptation Rules* consists of three parts: *event* specifies triggers for the adaptation, *condition*s need to be fulfilled to carry out the adaptation, and *action* defines the adaptation techniques to be applied. Additionally, every rule has a unique *id* and a *priority* to define the processing order.

Events define the triggers for adaptation, which may not only be published by application components, but by arbitrary parts of the composition environment, as well. *Application events* are actuated by components of a CMA and usually signalize state changes. *Context events* are issued by the *Context Manager* and contain updates of context parameters. *Runtime events* are published by the composition environment, e. g., upon successful component integration. Thus, they can be used to realize error handling with the help of adaptation rules instead of imperative programming. Finally, *complex events* describe non-atomic event patterns, such as causal dependencies. This way, a certain sequence of events can be defined to trigger an adaptation.

Listing 1 shows an exemplary event part of an adaptation rule. The latter is evaluated whenever a change of the user's device's battery level is detected and propagated.

```
1   <contextEvent contextParam="/user:currentDevice/dev
        :state/dev:batteryLevel" />
```

Listing 1. Definition of a rule's trigger *event*

The execution of adaptation rules is subject to **Conditions**. These consist of *terms* that can be combined with the help of logic operators (`AND`, `OR`) and use binary comparison operators (>,>=,!=, etc.). Further operators are supported, such as `CONTAINS` or such useful for semantic context representations (`TYPE`, `ISA`). Terms contain literal values or refer to context or event parameters. A typical case for using the latter is the extraction of a menu item's name

 ISBN: 978-1-61208-156-4

from a corresponding `itemSelected` event. In case of more complex conditions, SPARQL [19] can be used.

Listing 2 shows a condition, which restricts the adaptation to mobile devices (lines 7–10) with a battery level below 30% (lines 3–6).

```
1  <condition>
2    <and>
3      <term operator="lt">
4        <contextParam>/user:currentDevice/dev:state/dev
             :batteryLevel</contextParam>
5        <literal> 0.3 </literal>
6      </term>
7      <term operator="eq">
8        <contextParam>/user:currentDevice/dev:isMobile
             </contextParam>
9        <literal> true </literal>
10     </term>
11   </and>
12 </condition>
```

Listing 2. Exemplary *condition* for a rule

Finally, rules contain **Actions** representing concrete adaptation techniques, which are processed following their priority in a descending order. We use a generic vocabulary for these actions, which is derived from the techniques presented in Section II and easily extensible. It includes *component actions* addressing one or multiple components (e. g., removeComponent, reconfigure, setVisibility), *channel actions* adapting communication relationships (e. g., subscribe, unsubscribe, removeChannel, fireEvent), and *complex actions* changing application-wide concepts (e. g., changeLayout, updateContext).

Listing 3 shows an exemplary component reconfiguration – in this case the property *map_city* is set to the user's current location (a city) referenced from the context model.

```
1 <componentAction priority="9" pointcut="sightmap">
2   <reconfigureComponent property="map_city">
3     <contextParam>/user:currentLocation/space:name</
          contextParam>
4   </reconfigureComponent>
5 </componentAction>
```

Listing 3. Adaptation action to reconfigure a component property

The availability and consistency of context data is ensured by several components forming the context management subsystem, which are discussed in the next section.

C. Context Monitoring and Management

In this section we discuss how context is monitored, stored, and made available to adaptation system.

1) ***Remote Context Management – CroCo:*** Due to its complexity, we externalize context management to a dedicated *Context Service*. Therefore, we use the ontology-based service CroCo [20], which allows arbitrary *context providers* to submit, and *context consumers* to request context data via specific service interfaces. Thus, it serves as a general context supplier for external context-aware applications and platforms, such as ours.

Within CroCo context data is represented by a generic ontology-based context model, which uses several sub-ontologies describing aspects like time, place, the user, and his device. With the help of *domain profiles*, this model can be extended with additional concepts and relationships particular for specific domains. For the scenario mentioned before, we use concepts such as `UserProfile` (related to a `Person`), `WebBrowserConfiguration` (related to a `WebBrowser` being part of a `UserProfile`), and `UserLocation` (a type of `City`).

By using CroCo, our adaptation architecture highly benefits from its sophisticated means to check the consistency of the context model, to detect conflicts, and to infer additional knowledge with the help of semantic reasoning. These mechanisms guarantee, that we can rely on a valid and comprehensive model at minimal cost. Context information can either by requested synchronously, or consumers register for specific data, so that they are notified once it changes.

More detailed information on its architecture, inner workings, and the ontology-based model can be found in [20], including several examples of use.

2) ***Local Context Management:*** Our concept entails a local (client-side) context management to stay independent from the context service used, and to improve performance. A slice of the context model relevant for the application is cached by the adaptation system. That way, not every context request must be sent to the remote service, but can be evaluated with the local model which is continuously synchronized in the background. The following components are responsible for context management within our system.

Context Monitors are software components of the adaptation system that sense context data. Once it changes, monitors publish context updates to the *Context Manager*. Internally, they process raw context data to comply with the context model used, e. g., by mapping it to semantic concepts of an ontology. In addition to the actual data, they can provide a *confidence* value as discussed in [20].

The behavior of monitors can be configured as part of the application model, e. g., to set the minimal time threshold between context updates, or to request user confirmation before context updates are published to CroCo. Monitors provided by our system itself need not cover all context parameters used by the application as arbitrary external context providers may as well contribute to the model. Further, by implementing a dedicated monitor interface, new sensors can easily be added.

The **Context Manager** provides a uniform interface to the context model, using the local and remote context store, transparently. Its central responsibility is the local management of context information to minimize synchronous requests to the *Context Service* for every single parameter needed. Therefore, it initially downloads all parameters required for component configurations and rule evaluations from CroCo and registers there to be notified in case of

 ISBN: 978-1-61208-156-4

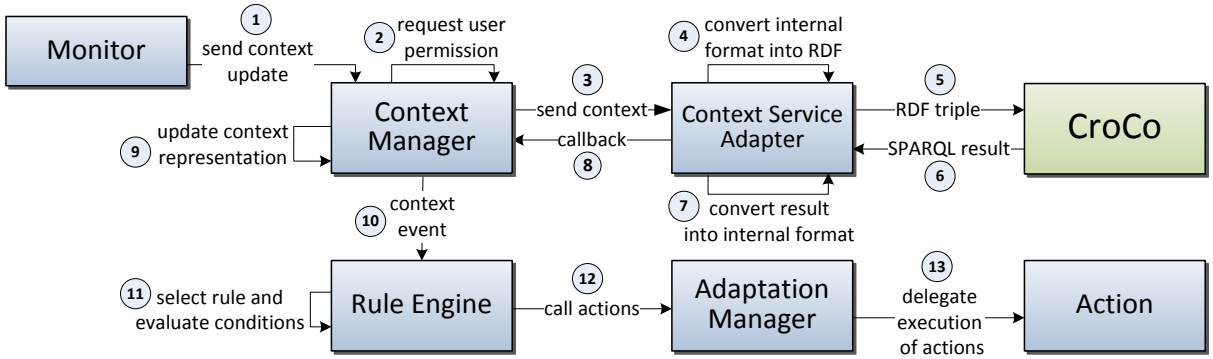


Figure 2. Workflow of the adaptation mechanism

model updates. Upon notification, the local model representation is synchronized with the remote model and context update events are issued, which trigger processing of the corresponding adaptation rules. Only on special occasions, e. g., when complex SPARQL requests need to be evaluated, requests are forwarded to CROCO.

Additionally, the *Context Manager* manages all local monitors and offers methods for (un)registering and (de)activating them. Their data is passed to CROCO for validation, consolidation, merging with the model and reasoning.

Communication with the external context management facility is handled by *Context Service Adapters* (CSA), which hide implementation and interface specifics of the context service used behind a uniform interface. Hence, it can be exchanged with an arbitrary local or remote solution, as long as a proper adapter exists. With regard to CROCO, the corresponding adapter transforms the context query and update events into RDF. Thereby, the implicit creation of instances and relations may be necessary. To accomplish this task, structural knowledge about the context model is needed, which can either be hard-coded into the adapter, or be provided dynamically by the context service, as is the case with our system.

D. Dynamic Adaptation Management

Figure 1 illustrates the modules involved in the dynamic adaptation process and their relation.

The **Adaptation Manager**'s main responsibility lies in providing and managing adaptation techniques. Whenever the evaluation of a rule implies a certain adaptation, the manager delegates its execution to the corresponding action implementation. It is also the central management entity within the adaptation system. Externally it offers a facade, i. e., an interface to request context parameters and process adaptation rules. Internally it is responsible for initializing and managing the other adaptation components. Before the application initialization, it loads configuration data and rules and subsequently initializes the *Rule Engine*, *Context Service Adapter* and *Context Manager*. The *Rule Engine* is provided with the necessary rules, and context parameter references from component configurations are replaced with the actual values requested from the *Context Manager*.

The **Rule Engine** is designed to evaluate adaptation rules. Therefore, it is automatically registered to be notified with all application and runtime events that are referenced by rules. Upon notification, the engine identifies all affected rules and sorts them with respect to their priority. Subsequently, their conditions are evaluated and the *Adaptation Manager* is called to execute the resulting actions.

Actions are platform-specific implementations of adaptation techniques as discussed above. For every abstract action there exists a corresponding implementation, which is naturally specific to the composition environment.

E. Adaptation Process

Figure 2 illustrates the steps of a dynamic adaptation and the components taking part in this process. Basically two very similar scenarios can be distinguished: In the simple case, context updates are sent from the external context service and result in an adaptation. In the extended case, updates result from data sensed by local monitors. Figure 2 presents the second case – the simple one is included starting with step six.

1. A monitor instructs the Context Manager to publish a detected context change.
2/3. The Context Manager checks, whether a user confirmation is required to publish such (possibly sensitive) data to the external context service. As discussed before, this can be configured as part of the adaptivity model. If a confirmation is needed, the data is presented to the user to be approved. Upon confirmation, the updated context data is handed to the CSA.
4/5. The latter parses the data, generates a representation particular to the context service and sends it there. Since we use CROCO, RDF triples are built to conform with its service interface.
6. CROCO carries out consistency checks, validation, and reasoning of context data. Then, context consumers registered with the affected data are notified. Consequently, the CSA is informed that parameters relevant for the evaluation of adaptation rules have changed.

 ISBN: 978-1-61208-156-4

7/8. After parsing and transforming the updated context data into the internal representation, the CSA forwards it to the Context Manager.

9/10. The latter updates its local representation of the context model and informs the Rule Engine about model changes using the corresponding context events.

11. The Rule Engine analyzes the events, searches for adaptation rules that apply to the updated context parameters, and evaluates their conditions. If additional context parameters are referenced, those are obtained with the help of the Context Manager.

12. Finally, the Rule Engine returns a list of adaptation actions to be executed to the Adaptation Manager.

13. The latter then chooses a suitable action implementation, triggers and coordinates its execution.

Even though the context information coming from monitors could be directly merged into the local model, it is instead first sent to the context service. This is for the obvious reason that CROCO employs mechanisms for validation, consistency checking and reasoning which determine whether the information are merged into the model or not. Furthermore, this round trip ensures that only the context data needed is processed by the Context Manager.

V. IMPLEMENTATION AND USAGE

Based on the concepts discussed we implemented a prototype of the adaptation system and integrated it with our composition platform CRUISe. In the following, we outline relevant parts of our implementation including an exemplary application realizing the travel guide scenario.

A. Integration of the adaptation system in CRUISe

The adaptation system was integrated with the CRUISe *Thin Server Runtime* (TSR), a browser-based composition environment. Its event-based communication allows for an easy integration by adding the Adaptation Manager as a publisher and subscriber to the global event bus. Several JavaScript-based monitors were implemented, e. g., to detect browser settings and the approximate location by means of the *Geolocation* API (http://dev.w3.org/geo/api/). The local context model is a JSON representation of CROCO's OWL-based model. Communication with CROCO is realized via SOAP using a dedicated adapter that generates SPARQL queries for synchronous requests. Asynchronous context notifications rely on CometD (http://cometd.org/).

The adaptation rules are derived from the composition model and dynamically evaluated by the Rule Engine. The corresponding actions are realized in JavaScript, extending a universal interface. Adaptability is offered by a component panel (see Figure 3) with options to minimize, close, and exchange components with alternatives. The latter proved to be the most challenging due to the dynamic integration of new code and the state transfer realized using proxies and the memento pattern.

B. Sample Application *TravelGuide*

To validate our concept and implementation, we built the *TravelGuide* application (Fig. 3) based on the use cases discussed at the beginning of this paper. To realize the desired behavior, a number of components were implemented, including a Google Map (on the right, indicating sights around the user and calculating routes there), a details form (bottom left), and an image gallery for the sight (top left). As an alternative to the latter, a video viewer was built which fetches videos of the sight from YouTube and has the same interface as the gallery, hence both are exchangeable. All visible components can be removed, restored, and explicitly exchanged with alternatives using a drop down menu.

The adaptivity modeled includes context-sensitive configurations, e. g., with user's native language, and the following behavior: 1) The map is continuously synchronized with the user's location; 2) His interest in cultural activities leads to the display of optional sights, e. g., museums, on the map; 3) The overall layout depends on the available screen estate and dynamically switches between a single- and multi-column layout; 4) When the battery power of the user's device falls below 30%, the video player component is exchanged with the image gallery in case the device is mobile.

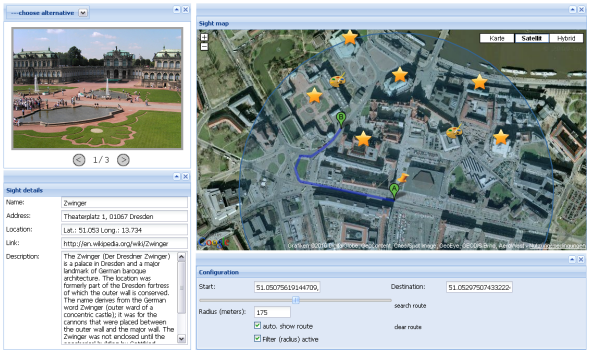


Figure 3. Adaptive composite *TravelGuide* application

Overall, our prototype proved to be stable and sufficiently supported all of the adaptation techniques defined, including the adaptation of components, their properties and the layout – both implicitly and explicitly by the user – as well as the dynamic adaptation of the control and data flow. Adaptation of component interfaces, i. e., mediation, was not explicitely modeled, as it is inherentely provided by the CRUISe platform.

VI. CONCLUSION AND FUTURE WORK

In this paper we present an adaptation concept which successfully transfers the principles of traditional *Adaptive Hypermedia* techniques to composite mashup applications. The definition of adaptive behavior is based on platform- and application-independent rules. They define dynamic adaptation techniques, such as component reconfiguration and exchange, data and control flow as well as adaptive

 ISBN: 978-1-61208-156-4

layout and screen flow at run time. To this end, the system provides context monitoring as well as context management entities. For the latter, a dedicated, ontology-based context management service is used, which can be transparently exchanged. A prototypical implementation of our solution and its integration with the CRUISe composition platform proved the feasibility and practicability of our concept.

In summary, we provide a valuable approach for the adaptation of presentation-oriented mashup applications. It combines a sophisticated context management with an efficient adaptation runtime, while keeping all means of context-awareness separated from the application. Thus, adaptation can be handled as an additional aspect. Our system can be integrated with other composition environments, provided that platform-specific adaptation actions are implemented.

One aspect that proved challenging was the definition of adaptive behavior. Even though the rules are easy to understand, the interactions between rules and the interactions with component-internal adaptivity are hard to overlook at design-time. Therefore, we are currently investigating on how to improve modeling and authoring of adaptive behavior. As part of this effort, we strive for a simplification and abstraction of the adaptivity model including higher-level *adaptation aspects* which take into consideration the semantics and self-adaptation capabilities of components. Furthermore, we are working on improving context management, by 1) letting CroCo autonomously learn and gather knowledge with the help of other services and 2) incorporating a better deduction of context from user interactions.

VII. Acknowledgements

The CRUISe project was funded by the BMBF under promotional reference number 01IS08034-C. The work of Carsten Radeck is funded by the ESF and Free State Saxony (Germany) under reference number ESF-080951805.

References

[1] S. Pietschmann, T. Nestler, and F. Daniel, "Application Composition at the Presentation Layer: Alternatives and Open Issues," in *Proc. of the Intl. Conf. on Information Integration and Web-based Applications & Services*, 2010.

[2] F. Daniel, F. Casati, B. Benatallah, and M.-C. Shan, "Hosted Universal Composition: Models, Languages and Infrastructure in mashArt," in *Proc. of the 28th Intl. Conf. on Conceptual Modeling*, November 2009.

[3] S. Pietschmann, "A Model-Driven Development Process and Runtime Platform for Adaptive Composite Web Applications," *Intl. Journal on Advances in Internet Technology*, vol. 4, 2010.

[4] P. Brusilovsky and N. Henze, "Open Corpus Adaptive Educational Hypermedia," in *Adaptive Web: Methods and Strategies of Web Personalization*, vol. 4321, 2007, pp. 671–696.

[5] D. Benslimane, S. Dustdar, and A. Sheth, "Service Mashups," *Internet Computing*, vol. 12, no. 5, pp. 13–15, 2008.

[6] S. Pietschmann, V. Tietz, J. Reimann, C. Liebing, M. Pohle, and K. Meißner, "A Metamodel for Context- Aware Component-Based Mashup Applications," in *Proc. of the Intl. Conf. on Information Integration and Web-based Applications & Services*, 2010.

[7] D. Lizcano, J. Soriano, M. Reyes, and J. J. Hierro, "EzWeb/FAST: Reporting on a successful mashup-based solution for developing and deploying composite applications in the upcoming web of services," in *Proc. of the 10th Intl. Conf. on Information Integration and Web-based Applications & Services*. New York, NY, USA: ACM, 2008, pp. 15–24.

[8] A. Erradi, P. Maheshwari, and V. Tosic, "Policy-Driven Middleware for Self-adaptation of Web Services Compositions," in *Proc. of the Intl. Conf. on Middleware*, 2006, pp. 62–80.

[9] P. Brusilovsky, "Adaptive Hypermedia," *User Modeling and User-Adapted Interaction*, vol. 11, no. 1-2, pp. 87–110, 2001.

[10] A. Ketfi, N. Belkhatir, and P.-Y. Cunin, "Automatic Adaptation of Component-based Software: Issues and Experiences," in *Proc. of the Intl. Conf. on Parallel and Distributed Processing Techniques and Applications*, 2002, pp. 1365–1371.

[11] K. Geihs, P. Barone, F. Eliassen, J. Floch, R. Fricke, E. Gjorven, S. Hallsteinsen, G. Horn, M. U. Khan, A. Mamelli, G. A. Papadopoulos, N. Paspallis, R. Reichle, and E. Stav, "A comprehensive solution for application-level adaptation," *Software – Practice & Experience*, vol. 39, no. 4, 2009.

[12] M. Hinz, "Kontextsensitive Generierung adaptiver multimedialer Webanwendungen," Ph.D. dissertation, Technische Universität Dresden, 2008.

[13] K.-U. Schmidt, R. Stühmer, and L. Stojanovic, "Gaining Reactivity for Rich Internet Applications by Introducing Client-side Complex Event Processing and Declarative Rules," in *Proc. of the AAAI Spring Symposium on Intelligent Event Processing*, 2009, pp. 67–72.

[14] N. Gui, V. Florio, H. Sun, and C. Blondia, "ACCADA: A Framework for Continuous Context-Aware Deployment and Adaptation," in *Proc. of the 11th Intl. Symposium on Stabilization, Safety, and Security of Distributed Systems*, 2009, pp. 325–340.

[15] F. André, E. Daubert, and G. Gauvrit, "Towards a Generic Context-Aware Framework for Self- Adaptation of Service-Oriented Architectures," in *Prof. of the 5th Intl. Conf. on Internet and Web Applications and Services*, 2010.

[16] Q. Z. Sheng, B. Benatallah, Z. Maamar, and A. H. Ngu, "Configurable Composition and Adaptive Provisioning of Web Services," *IEEE Transactions on Services Computing (TSC)*, vol. 2, no. 1, pp. 34–49, 2009.

[17] T. Fischer, F. Bakalov, and A. Nauerz, "Towards an Automatic Service Composition for Generation of User-Sensitive Mashups," 2008.

[18] F. Daniel and M. Matera, "Mashing Up Context-Aware Web Applications: A Component-Based Development Approach," in *Proc. of the Intl. Conf. on Web Informa- tion Systems Engineering*, 2008, pp. 250–263.

[19] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," 2008, http://www.w3.org/TR/rdf-sparql-query/.

[20] A. Mitschick, S. Pietschmann, and K. Meißner, "An Ontology-Based, Cross-Application Context Modeling and Management Service," *Intl. Journal on Semantic Web and Information Systems*, Feb. 2010.

 ISBN: 978-1-61208-156-4

Input-adaptive QMC-Kalman filters for track fitting

Rodolfo G. Esteves and Michael D. McCool
School of Computer Science
University of Waterloo
{*rgesteve, mmccool*}*@cs.uwaterloo.ca*

Christiane Lemieux
Department of Statistics and Actuarial Science
University of Waterloo
clemieux@math.uwaterloo.ca

***Abstract*—On-line track reconstruction is one of the bottlenecks of the pattern recognition task in High-Energy Physics (HEP). This problem has been traditionally divided into the sub-tasks of track finding and track fitting. The latter involves estimating the state of a particle inside a detector moving under the influence of a magnetic field. For the last twenty or so years the most popular solution to the track fitting problem has been the Kalman filter (KF). It is well known that the KF is only guaranteed to compute the optimal estimator if the dynamics of the system are linear and subject to Gaussian noise. However, these conditions are not met in the track fitting problem, in particular, the dynamics are strongly non-Gaussian due to effects such as multiple Coulomb scattering and energy loss. A proposed solution is the "Gaussian sum filter" (GSF) which runs a bank of KFs to estimate each of the modes of the noise distributions, modelled here as a mixture of Gaussians. But this solution is limited by the fact that the GSF uses the same distribution for every input dataset. To address this issue, we present in this paper the Input-adaptive KF (IAKF), which makes use of the dynamic code generation features in Intel's ArBB parallel framework to create a GSF that matches the given (observation) noise distribution. The IAKF further deals with non-linearity by having the GSF drive, instead of KFs, the recently proposed Quasi-Monte Carlo KF (QMC-KF), a generalization of the σ-point KF family. Numerical results are shown to validate the performance of the IAKF. The generated code is not only tailored to the data, but takes advantage of several levels of parallelism in multi-core processors.**

***Keywords*-Nonlinear dynamic systems; quasi-Monte Carlo (QMC); Kalman filter (KF)**

I. INTRODUCTION

High-Energy Physics (HEP) studies the fundamental components of matter and radiation, as well as their interactions, thereby addressing questions crucial for the understanding of the Universe. HEP experiments pose unique challenges in design, implementation and data analysis. For one, HEP experiments often have to sort through the massive data streams produced by particle accelerators. Particle accelerators use electromagnetic fields to induce high-momenta on sub-atomic particles, whose collisions among each other generate data that is invaluable to understand matter under extreme conditions. This data often takes the form of *traces*, measurements of various physical aspects of the particle taken along their collision paths at specific detection points (the "stations"). A considerable portion of the data mining that takes place in a HEP experiment is spent in the *track reconstruction* task, which consists of taking the traces and reconstructing the underlying physical process. Track reconstruction is often broken up in the complementary sub-tasks of *track finding* and *track fitting*. Track finding involves associating a set of readings with the likely trajectory of a specific particle. Track fitting then takes those sets and determines the values that best conciliate the experimental readings and the mathematical description of the trajectory. These values, or *state* are usually denoted by $\mathbf{x}_k \in \mathbb{R}^5$, and consist of the exact intersection point of the particle with each of the detectors, the track direction and the curvature. A *measurement* $\mathbf{z}_k$ is taken at each station k, and a collection of stations represents a model of the whole detector.

The mathematical setting of this problem is to consider the particle accelerator as a nonlinear dynamic system [1]:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \omega_k \tag{1}$$

of which the measurement $\mathbf{z}_k$ is a function, also polluted by noise to form the stochastic "observation" process:

$$\mathbf{z}_k = h(\mathbf{x}_k) + \nu_k \tag{2}$$

It can be seen from the notation that the system is considered as discretized in time with indices k. In the transition between measurement points $k-1$ and k, the state is considered to be subject to *process noise*, which is denoted here by ω_k. Moreover, precise observations are not possible because of the limitations in the measuring model and instruments, a circumstance which is considered by introducing the *measurement noise* ν_k. The precise probability distributions of ω_k and ν_k heavily depend on the application, but they are commonly taken to be mutually independent.

The goal of track fitting is then to *filter* out the noise, i.e., to calculate the posterior distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ at all measurement indices k, where $\mathbf{z}_{1:k}$ denotes a sequence of observations $\{\mathbf{z}_i\}_{i=1}^{k}$. The optimal solution to this problem is given by the recursive Bayesian estimation algorithm [2], which recursively updates the posterior density of the system state as new observations arrive. However, it is worth noting that this recursive solution is only tractable for linear, Gaussian systems, in which case the closed-form recursive solution to the Bayesian integral equations is the well-known Kalman filter (KF). Because in HEP experiments process noise arises from interactions between charged particles and

 ISBN: 978-1-61208-156-4

detector materials, the linearity and Gaussianity assumptions the KF relies on are hardly ever met. The measurement noise v_k is also rarely Gaussian, since in real detectors there is an ever-present possibility of outlying or ambiguous observations. Therefore, approximate solutions need to be applied to this problem.

Approximate filtering for general dynamic state space systems can be roughly categorized in two approaches: deterministic and Sequential Monte Carlo-based. The first approach, which is the focus of this paper, generalizes the KF, trying to keep its simplicity and well-understood theory [1], [2], [3]. However, these generalizations suffer from the fact that filters must be manually constructed from the input data, or that the same filter is applied to every input dataset, which degrades the quality of the estimator or require inordinate user effort. To overcome these disadvantages, we present the Input-adaptive KF (IAKF), an approximate solution to the filtering problem based on the KF where an automatically constructed Gaussian mixture models the measurement noise and a set of non-linear KFs are used for the actual filtering. We note that some ideas preliminary to this paper have been presented in [4], but a paper-length exposition has never been published.

The organization of this paper is as follows: Section II introduces the variants of the KF designed to handle non-linear dynamics in the presence of non-Gaussian noise. Our proposed method, the IAKF is presented in Section III, where we describe its operation and demonstrate its statistical performance. The most novel feature of the IAKF is that it is implemented by relying heavily on dynamic code generation, an aspect we explore in section IV. After a brief introduction to this technology, we present the dynamic code generation facilities of ArBB, and how we use them to implement the IAKF. We also compare previous approaches to code generation for filtering with our work. Finally, we conclude in Section V.

II. Non-linear generalizations to the KF

The KF is a prediction/correction scheme with the predicted state and observation being calculated from the estimate at the previous measurement point (or from the prior distribution $p(x_0)$):

$$\begin{aligned} \mathbf{x}_{k|k-1} &= f(\mathbf{x}_{k-1|k-1}, \omega_k) \\ \mathbf{z}_{k|k-1} &= h(\mathbf{x}_{k|k-1}, v_k) \end{aligned}$$

where ω_k and v_k are independent and normally-distributed, with zero mean and covariances Q and R respectively. f is a stochastic transition kernel from state $\mathbf{x}_{k-1}$ to state $\mathbf{x}_k$ and h is a stochastic non-linear mapping from the current state to the observation. $\mathbf{x}_{k-1|k-1}$ is the state estimate given the observations $\mathbf{z}_{1:k-1}$, and $\mathbf{x}_{k|k-1}$ is the *predicted* estimate for the next state given the same trace.

These predictions are then updated with the *innovation*, the difference between predicted and actually observed measurement, modulated with a correcting factor K_k, the *Kalman gain*, to render the next state estimate $\mathbf{x}_{k|k}$ and corresponding covariance $\mathbf{P}_{k|k}$:

$$\begin{aligned} \mathbf{x}_{k|k} &= \mathbf{x}_{k|k-1} - K_k(\mathbf{z}_k - \mathbf{z}_{k|k-1}) \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - K_k(P_z)_{k-1}K_k^T \end{aligned}$$

The KF is *optimal* (in the least-squares sense) provided process and measurement mappings are linear, and the associated noise is Gaussian. However, as mentioned in Section I, this is not the case in HEP experiments. Below, we present two general approaches dealing with non-linearities and non-Gaussianity, particularly relevant to track fitting.

A. Dealing with non-linearities

When considering non-linearities, a common simplifying assumption is that

$$\begin{aligned} p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) &= \mathcal{N}(\mathbf{x}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}) \\ p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) &= \mathcal{N}(\mathbf{x}_{k|k-1}, \mathbf{P}_{k|k-1}) \end{aligned}$$

which makes the filtering distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ normally distributed as well.

The Extended Kalman filter (EKF) is by far the most popular technique for state estimation on non-linear dynamic systems. Despite its advantages, the EKF comes at the cost of considerable shortcomings, most of which result from the EKF's linearization of process and measurement equations around the previous estimate. This does not take into account the statistical properties of the noise, and may ultimately cause the divergence of the filter [1].

Alternative approaches go back to the definition of first- and second-moment of the filtering distribution:

$$\begin{aligned} \mathbf{x}_{k|k-1} &= \int \mathbf{x}_k p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_k \\ &= \int f(\mathbf{x}_{k-1}) p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1}) \mathrm{d}\mathbf{x}_{k-1} \end{aligned} \quad (3)$$

The second moment follows the same pattern and is ommitted here due to space restrictions.

In this paper we present the Input-adaptive KF, which makes use of the QMC-KF algorithm to construct filters that are tailored to the input data, thereby improving the robustness of a track-fitting system as shown in Section III-B. The QMC-KF numerically approximates (3) using Monte Carlo or quasi-Monte Carlo (QMC) integration. QMC-KF [2], a generalization of σ-point filters, relies on the approximation

$$\mathbf{x}_{k|k-1} = \sum_{i=1}^{n} f(\mathbf{x}_k^{(i)}) \quad (4)$$

where $[\mathbf{x}_k^{(i)}]_{i=1}^n$ is a low-discrepancy point set of the appropriate dimensionality (in this work, as in [2], we use randomized Halton point sets) under some transformation that maps to a Gaussian distribution. This can be done in several ways, and we explain our choices in Section III-B.

 ISBN: 978-1-61208-156-4

B. Dealing with non-Gaussianity

Gaussian distributions are hardly appropriate for the phenomena studied in HEP experiments. Measurements include outliers and ambiguity that introduce tails in the measurement error distribution v_k, whereas the biggest contributors to process noise, energy loss and multiple scattering, are highly non-Gaussian. Forcing a Gaussian distribution to describe these effects greatly reduces the amount of information contained in the true densities, especially in the case of multimodal densities.

A common approach to avoid information loss while remaining within the KF framework consists of modelling the non-Gaussian distributions by *Gaussian mixtures*. For example, measurement outliers can be handled by a Gaussian mixture with a "core" component describing the "regular" measurements, and one or more components describing the outlier-induced tails [5] (e.g., by a mixture of Gaussians sharing the mean but with different covariances). Likewise, ambiguous measurements can be modeled by a mixture of Gaussians with one component per possible value, i.e., with the mean set to the possible value and identical variances, thus concurrently using all possible meanings of the ambiguous measurement. As for process noise, we can model the tails of the multiple scattering for low-energy particles, or the highly asymmetric energy loss of electrons by suitable Gaussian sums [6]. In principle, every distribution involved in the filtering process (state priors, measurement and process noise) can be modeled as a Gaussian mixture. Taking this notion as a guideline, Alspach and Sorenson [3] proposed the Gaussian-sum filter (GSF), where every component of the mixture is propagated and updated by a standard KF. This is to say that the GSF consists of a bank of Kalman filters running in parallel.

III. The Input-adaptive Kalman filter

All the techniques described in the above section assume that the same filter will be applied to every input, since parameters chosen for the filter are fixed at program construction time. This does not reflect the actual system under consideration, and may therefore result in inaccurate or divergent filters. Here we address this issue by dynamically building the filter based on the input data. In this section we present the Input-adaptive KF (IAKF), a GSF driving QMC-KFs with a preprocessing step of data clustering.

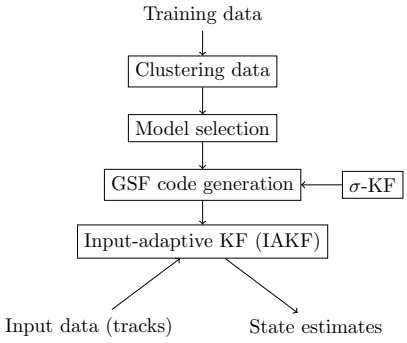


Figure 1. Conceptual diagram of the IAKF

A. Operation of the IAKF

Figure 1. shows the conceptual diagram of the IAKF, whose general operation is as follows: a sample of the tracks at the first station is read as a training set, assumed to have been generated by a Gaussian mixture density

$$p(x_0) = \sum_{i=1}^{l} \alpha_i \mathcal{N}(\mu_i, C_i) \tag{5}$$

We fit the training data into the Gaussian mixture, which models the notion that the measurement noise is multimodal due to outliers. It is premature to decide *a priori* how many terms l the mixture is to have because of the lack of information on the outlier distribution. Therefore we run a parameter-estimation task on several candidate component counts and use a model selection criterion to choose the best fit among them. The parameter estimation procedure determines the values for mixture parameters and proportions. Having determined the appropriate number of components, we generate the corresponding QMC-KF instances, which will be "baked in" the final filter. Finally that resulting filter is run on the input data to perform the actual state estimation. It is worth noting that the mixing proportion of the Gaussian sum needs to be reweighted at every iteration to maintain an accurate estimate.

The model selection criteria we use to determine how many components the Gaussian mixture should have is the Bayesian information criterion (BIC). For parameter estimation, we have chosen the Expectation Maximization (EM) algorithm, a commonly used maximum-likelihood technique to estimate the parameters of a distribution of a specified form from training data. As the computational load of the GSF is directly related to the number of components in the mixture, we run EM for a low component count (2 to 5).

In the GSF, care must be taken to control the possible combinatorial explosion of Gaussian terms in the posteriors. As the IAKF has a fixed number of components, when any of the constituent QMC-KF estimates becomes multimodal, the smaller components are dropped as to maintain a constant component count.

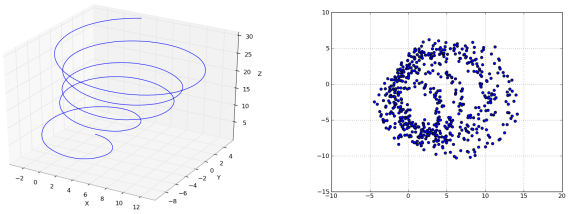
B. Numerical evaluation

To validate the computational performance of the IAKF implementation, we test our system on simulated data. We use the methodology proposed in [7] to simulate the transit of a charged particle in a magnetic field. Therefore, the function f takes the form of a 4th-order Runge-Kutta solution (with fixed-step size) to the equation

$$d\mathbf{p} = \kappa q(\mathbf{v} \times \mathbf{B}) ds/|\mathbf{v}|$$

 ISBN: 978-1-61208-156-4

where $\mathbf{p}$ is the momentum of the particle, q its charge, $\mathbf{v}$ its velocity, $dt = ds/|\mathbf{v}|$ the trajectory length, $\mathbf{B}$ the magnetic field, and κ is a constant. The state is the 5-tuple $(x, y, t_x, t_y, q/|\mathbf{p}|)$ where $(x, y)_z$ denote the intersection of the trajectory with detecting surface z, $t_x(z) = dx/dz$ and $t_y(z) = dy/dz$ indicate the particle's direction at that point. The observation function h mimics the way silicon micro-strip detectors carry out measurements, projecting the x, y-coordinates at the intersections with the stations z. An illustration of a simplified version of the above system is shown in Figure 2. The simplifications consist of using an homogeneous magnetic field and a (uni-modal) Gaussian noise for both system (a) and measurement (b) noise.



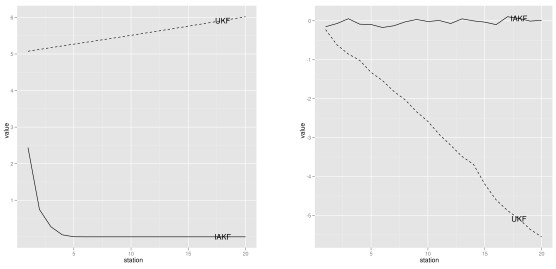
(a) Dynamics process f (motion of charged particle in a magnetic field)

(b) Measurement process h

Figure 2. Non-linear functions f and h for track fitting

The actual experimental setup consists of adding 2-, 3- and 4-component Gaussians mixture additive noise to the measurement process on a 7-station detector. The dynamics of the system are encoded in the Runge-Kutta solver described above.

We test three filters: the Unscented Kalman filter (UKF) and the IAKF driving both QMC-KFs and traditional Monte Carlo filters (i.e., a non-linear filter making use of Monte Carlo integration of the Kalman filter recurrences). Our QMC-KF filters are fed by a 1024-randomized Halton sequence of state dimension 5. The particular randomization method is to use an Owen-type scrambling [8].



(a) Covariance trace

(b) State estimation residuals

Figure 3. Performance of 2-component IAKF (solid) vs UKF (dashed)

Figure 3 illustrates the relative filter efficiency of the UKF vs the 2-component IAKF/QMC on simulated runs. It averages the RMSE (i.e., the trace of the covariance matrix $\mathbf{P}_k$) on 1024 tracks on 20 stations subject to a 2-component Gaussian mixture "outlier" measurement noise (i.e., two components with the same mean but different covariance). It is shown that the IAKF (solid line) presents a significantly smaller and steadier filtering error than the UKF (dashed line), thus providing more accurate and stable state estimation.

IV. Run-time code generation of the IAKF

When developing a program, there is a constant tension between information and time: often, decisions about a program structure have to be made and implemented well before any input is actually seen; when input data becomes available to the program, it is too late for the characteristics of that data to shape the program's execution in a deep level. A system that could interpret the specific input and *generate* code specifically tailored to that input and the host environment would see improved performance. Such a system clearly has the need to programmatically manipulate program code. This is made easier by a simple, regular syntax, so systems based on Lisp or its variants are highly favoured in this type of application. A common solution to allow the creation and execution of code at run-time in languages with richer syntax is to provide a special function (usually called `eval`) that takes a string containing the source code to be executed, and executes it immediately. Incrementally building strings that constitute correct and complete programs is difficult enough, but having to deal with dependencies, variables and other bookkeeping that allows communication between host and created program constitutes a cost that far outweighs the possible benefits.

A more tractable solution has been found to use a two-stage approach, where the code to be generated is restricted to a small domain-specific language (DSL). Instead of feeding text written in this language to an `eval`-like function, a full language processing infrastructure processes the DSL source and somehow injects it to the main code, which is written in a general-purpose language. The best-known example of this form of DSL is the pair `lex`/`yacc` for lexer/parser generation, which generates code from the DSL at compile-time. The recent proliferation of just-in-time (JIT) compilers has made this two-stage approach possible at run-time, but the maintenance of two separate source translation infrastructures is still cumbersome. Recent work addresses this problem by using an "embedded" variation of DSL, which uses some facilities of a general-purpose language, to *host* a more specific language. Examples of this technique can be found in a number of modern languages, like the Spirit [9] parser generator C++ library, the LINQ language-integrated data-query framework for Microsoft's .NET, and most famously, numerous examples in the Ruby language, like Ruby on Rails. Embedded domain-specific languages (EDSLs) and JIT compilers are a good fit, as it is

 ISBN: 978-1-61208-156-4

much easier to develop a compiler for a simple language than to take on the task of developing an incremental compiler for a full language.

A central issue when developing EDSLs is the choice of data structure to represent programs or program fragments. As stated above, the most natural to use is a string containing the source of the program to be executed at the next stage. This representation, however, is rife with problems, as the contents of a string are effectively out of the control of the programming language. An alternative is the macro-like approach incorporated in the Intel Array Building Blocks (ArBB) library, which is used in this paper and will be described next.

A. Code generation in ArBB

ArBB, formerly Ct, library [10] is a retargetable dynamic compilation framework whose main purpose is to facilitate the coding of programs that are to take advantage of modern multi- and many-core architectures. As such, it provides a set of implicitly data-parallel collection data structures and computational patterns (including *map*, *reduce* and *prefix sum*). A programmer can make use of this abstract notation and target several levels of parallelism present in multi-core homogeneous systems, as well as potentially heterogeneous accelerator-based many-core architectures (for example, Graphics Processing Units) and cluster-based systems.

A more complete description of ArBB is best found elsewhere [11]. Here, we focus on the code-generating aspect. ArBB is a two-stage system, where the programmer works within the usual confines of the C++ language (the *uncaptured* environment), but specifies the functions that are to be run in parallel using library-provided data types, functions and control flow (the *captured* environment). The rationale for this two-level architecture is that captured code can be compiled to make maximum use of the system it is running on: vectorized ALUs, multiple processors, accelerators and other possible system configurations.

ArBB provides mirrors of C++ control structures in the form of the 'keywords' `_if`, `_for`, `_while` and `_do`. This inclusion allows the programmer to express the effect of the computation serially, even though this computation will ultimately be performed in parallel. For example, the kernel:

```
void nr_sqrt_kernel(arbb::f32 a, arbb::f32& s)
{
    _if (a < 0) { return -1; } _end_if;
    arbb::f32 sprev = arbb::f32(0);
    s = a;

    _while (arbb::abs(xprev -x) > tol) {
        sprev = s
        s = 0.5 * (sprev + a / sprev)
    } _end_while;
}
```

can be the argument of a `map` operator to find the square root of each (positive) element in an ArBB array. Each invocation of `nr_sqrt_kernel` will execute concurrently.

An interesting side effect of this two-level execution approach can be seen when C++ control flow is used in *captured* mode. The following code:

```
unsigned int unroll_factor = 4;

void fill_vec_kernel(arbb::dense<arbb::f32>& v,
    arbb::f32 val) {
 for(size_t i=0; i < unroll_factor; i++)
    { v[i]=val; }
}
```

has the same effect as unrolling the loop:

```
void fill_vec_kernel(arbb::dense<arbb::f32>& v,
    arbb::f32 val)
{
 v[0] = val; v[1] = val; v[2] = val; v[3] = val;
}
```

In what follows, we will use this sort of template-based generation to produce QMC-KFs customised to the input. It is pertinent to note that, while *C++ templates* are one form of code generation and programmatic manipulation (known as *metaprogramming*), they are an altogether different mechanism from the one described above for ArBB. C++ template-based metaprogramming is a *compile-time* construct that can be used *in addition* to the run-time facilities we have explained.

B. The ArBB implementation of the IAKF

We have built an ArBB implementation of the IAKF, which determines the number of components `NUM_KF` by the procedure described in Section III and uses this number to generate as many QMC-GSF as necessary, in a manner sketched by:

```
for k = 0; k < NUM_KF; k++) {
  _for( i = N - 1, i >= 0, i-- ){
    // ... magnetic field setup ...
    filter(ts,ss,xInfo, ts.hitsX2.row(i), w,T,C);
    filter(ts,ss,yInfo, ts.hitsY2.row(i), w,T,C);
    for( int j = 0; j < 3; j++ ){
      H2[j] = H1[j];
      H1[j] = H0[j];
    }
    z2 = z1; z1 = z0;
  } _end_for;
}
```

The `filter` kernels are wrapper functions over `maps`, which result in a parallelization strategy both over tracks and over concurrent QMC-KFs. Below we present the results of some numerical experiments. As an experiment we ran the filter with 1- and 2-component mixtures to assess the computational load.

 ISBN: 978-1-61208-156-4

Figure 4 illustrates a benchmark of our system, running on a dual-Core Intel i3 at 2.27 GHz. It can be seen from that the running time is roughly proportional to the number of components. Beyond that, it is more interesting to note that the program is *scalable*, i.e., that the running time consistently decreases as the number of cores increases. This is greatly desirable, as core count is only likely to increase given the current hardware trends.

C. Other approaches to KF code generation

The automatic generation of data analysis programs has been explored for at least a decade. For example, the AutoBayes program synthesis system [12] generates C++ code from a declarative specification of the statistical model via deductive synthesis directed by code templates. As such, it most resembles the `lex`/`yacc`-approach of DSLs described above, where a separate programming language infrastructure is required.

Another example of a system that synthesizes KF and variants code is AutoFILTER [13], which similarly to AutoBayes, outputs C++ code from a textual specification specialized to the description of noise distributions and differential equations. An interesting variation that AutoFILTER includes is that it links against the libraries from the Octave linear algebra system, within which its output is supposed to be used. This illustrates another aspect of code generation within C++, its interoperability.

In contrast to the above-mentioned systems, our work only requires a standard C++ compiler. Furthermore, our generated code takes full advantage of the parallel features of the architecture the program is running on. On the other hand, significant work has gone into the automated verification and certification of the code generated by AutoBayes, an aspect of considerable importance, which is not covered in this paper. The interoperability aspect of AutoFILTER is also a straightforward addition to our system.

V. Conclusion

In this work we proposed the Input-adaptive Kalman filter (IAKF), a member of the deterministic, approximate, nonlinear filter family. In contrast with traditional methods, the IAKF adapts to the input, running as many filters as necessary to best fit the input data. This feature, validated by numerical results, makes its estimates more accurate. Furthermore, the IAKF is more robust to changes in data than its non-adaptive counterparts. To implement the IAKF, we make use of the run-time code generation and compilation afforded us by modern parallelism libraries. It is our contention that furnishing the end-programmer with the ability to tailor the program in data-driven programs, as inferencing systems must be, allows for simple and straightforward implementation of programs that deal better with realistic scenarios. Moreover, the improved running time that modern parallel hardware offers can be put to good use in more realistic models and greater variety of data.

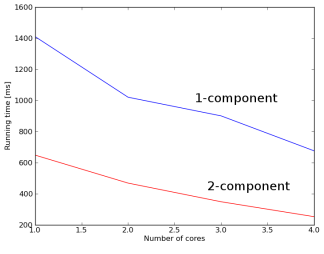


Figure 4. Running time of the QSF with 1- (in blue) and 2-component (in red) MoG ν_k

References

[1] R. van der Merwe and E. Wan, "Sigma-point Kalman filters for probabilistic inference in dynamic state-space models," in *Workshop on Advances in Machine Learning*, 2003.

[2] D. Guo and X. Wang, "Quasi-Monte Carlo filtering in nonlinear dynamic systems," *IEEE Transactions on signal processing*, vol. 54, 2006.

[3] D. L. Alspach and H. W. Sorenson, "Nonlinear Bayesian estimation using Gaussian sum approximations," *IEEE Transactions on automatic control*, vol. 17, no. 4, 1972.

[4] R. G. Esteves, C. Lemieux, and M. D. McCool, "Run-time generation of qmc-kalman filters for track fitting (abstract)," in *Monte Carlo and Quasi-Monte Carlo methods*, 2010.

[5] R. Frühwirth, "Track fitting with long-tailed noise: a Bayesian approach," *Computer Physics Communications*, vol. 85, 1995.

[6] W. Adam, R. Frühwirth, A. Strandlie, and T. Todorov, "Reconstruction of electron tracks with the Gaussian-sum filter," CERN, Tech. Rep. CERN-CMS-RN-2003-001, 2003.

[7] S. Gorbunov and I. Kisel, "An analytic formula for track extrapolation in an inhomogeneous magnetic field," in *International Workshop on Advanced Computing and Analysis Techniques in Physics Research*, 2005.

[8] C. Lemieux, *Monte Carlo and quasi-Monte Carlo Sampling*. Springer, 2009.

[9] J. de Guzman and H. Kaiser, "Boost.Spirit homepage," http://boost-spirit.com, 2010.

[10] "Intel Array Building Blocks homepage," http://software.intel.com/en-us/articles/intel-array-building-blocks, 2011.

[11] M. McCool, "Intel Array Building Blocks: A retargetable, dynamic compiler and embedded language," in *Code Generation and Optimization*, 2011.

[12] B. Fischer, J. Schumann, and T. Pressburger, "Generating data analysis programs from statistical models," in *Workshop on Semantics, Applications, and Implementation of Program Generation*, 2000.

[13] J. Richardson and E. Wilson, "Flexible generation of Kalman filter code," in *IEEE Aerospace Conference*, 2006.

 ISBN: 978-1-61208-156-4

An Adaptable Process Planning Tool

A Tool for Information, Comunication, and Interaction in a Robot Cell

Fredrik Danielsson
University West
Trollhättan, Sweden
e-mail: fredrik.danielsson@hv.se

Linn Gustavsson Christiernin
University West
Trollhättan, Sweden
e-mail: linn.gustavsson@hv.se

***Abstract*—This study presents work in progress on how to develop a process-planning tool to handle interaction between human operators and robots within a robot cell. First, we introduce how to include human activities in the process flow; then, we turn to our ideas for communication and feedback systems inside a robot cell. A small example of how to design interactive and re-programmable screens is presented.**

Keywords-Process planning tool; robot cells; automation; interaction; human operators and interface design.

I. INTRODUCTION

When producing automotive vehicles, a large part of the manufacturing process is automated. The vehicle parts go through different stations or cells to be adjusted and assembled and robots and humans have different, usually isolated, assignments. A typical robot cell might consist of a couple of robots up to as many as 20 robots. The entire cell is controlled and coordinated by a PLC (Programmable Logic Controller) or an industrial computer. A robot might be assigned a specific manufacturing process or to transport the product through the cell.

A problem with this type of automated production is the time it takes to make changes. An update in the vehicle model or the introduction of a new vehicle in the manufacturing line could take up to a year or more from planning to implementation [9]. The goal with our work is therefore a quicker process with fewer steps involved and production cells with robots easy to reconfigure and adapt so the time for changeover can be reduced. As a first step to try to achieve this goal, a process planning tool, P-SOP (Product oriented Sequence of Operation), was introduced in our earlier work [5] where the operators could create a new flow of activities for a robot cell through dragging and dropping icons in the tool. The tool then generated code from the graphical flow chart, which could be uploaded to the robots and PLCs. The next step to achieve more flexible production was to include the operator in the process; create possibilities for humans and robots to interact. The human operators should be able to go in to the cell and help the robot or interact with the robot to make the assembly more efficient. This required new functionality in our tool and a new way of thinking in the robot cell, involving safety, interaction, and efficiency issues.

This study will focus on how to reconfigure the software tool to be adaptable for human-machine interaction during the production. We will present our work in progress for how to 1) communicate to the robot that a part of an assignment is performed by a human operator, 2) give the human operator feedback on what to do and where and when to step in, 3) visualize the work flow, and 4) in real-time analyze the consequences of a human operator stepping in and performing a task. In this paper we will discuss how to change the software tool and how to create an adapting solution. The work is planned to be performed in two steps; firstly, create an adaptable tool, where the human interaction can be predetermined, and secondly, a fully adapting system where the human, on the fly, can go into the production line and assist the robot.

Next, we will introduce a brief background followed by our suggested solution and a small discussion of our work in progress.

II. BACKGROUND

This project started a few years back when the industry of Europe required quicker changeover times for changing automated production cells. Through a better and quicker process for how to give the robots instructions and reconfigure a cell a first step towards reduced changeover times could be achieved. A team was put together and worked within the FLEXA project [5, 7] to develop a tool and work with process development.

In the original process (also the process still used in most companies), a process planer creates a sketch or workflow for the activities a part should go through in a specific production cell. Information about the part, the material and different constraints are added to the sketch and the document is then sent to a subcontractor. They turn the sketch into instructions and code that is uploaded to the robots and PLC. They configure the cell to be “turn-key” ready for production. While this is done the process planer have sent work instructions to the operators of the cell for what to do and how. To speed up this process the idea is to exclude the subcontractor and create a tool where the process planner and operator can make the robots and PLC ready by themselves. Eventually the idea is to also exclude the process planer and only have an operator to perform the whole chain of activities.

A software tool was invented [5], where an operator could create new sequences or flows of instructions by combining graphical icons representing different tasks in the

 ISBN: 978-1-61208-156-4

cell. The sequence is translated into code and downloaded to the robots and PLC. This made it possible to put together different tasks for the robot to perform with rather short notice; creating a more flexible production.

Parallel to this project a second project called Flex Lean was initiated [6] where safety issues were addressed; due to safety regulations (HSG43 and EN ISO 13849-1) the humans are not allowed to be in the robot cell when active. If a human enters the robot cell during operation, sensors will activate the safety system. This will initiate the safety control system to cut power to the robots. Such a response will slow down production and halt the entire cell. In Flex Lean, these safety regulations have been further investigated and different ways for robots and humans to interact have been introduced within the safety regulations.

These two projects create the foundation for our work in this study. Where we add more functionality to the tool and assume that the safety issues of the cell can be solved through cooperative solutions for robots such as SafeMove from ABB [2, 12].

A. Related work

A more advanced process simulation tool and theories for how to best describe, model and simulate process planning activities are produced within the framework of FLEXA but by another research group [3, 19]. They have studied how to create and simulate the sequences in which tasks should be carried out but also how to include different constraints and organizational parameters when trying to optimize the process flow.

Comparing the more advanced tool to the current one in this study; the current tool tries to focus more on how to quickly re-program a robot cell and the actual interaction and communication language between operators and robots while the other tool focus more on a understanding of the process it self and the work model. Their tool is used as a guide earlier in the process when the actual steps of the process are being determined; they also try to visualize the workflow. In current work we assume that the process steps have been already determined by a tool like theirs or similar. We start with a process plan that has been simulated in earlier steps and then try to look at how to handle rapid changes in that sequence and how to, eventually, include the operator in the process sequence. We will however, in later work, try to merge the ideas from booth tools.

III. WORK IN PROGRESS - INTEGRATING HUMAN TASKS IN PROCESS FLOW

To create a situation where the human operator and the robots or machines can cooperate we need to adapt 1) the configuration tool, 2) the interface of communication, 3) the feedback system, and 4) the physical environment. In the current study we have initiated work on no. 1, 2 and 3 while 4 is considered in a parallel project and related studies. In our work we are very aware of the safety issues but also confident that results from related projects [2, 3, 6, 12, 19, 21] will solve the physical challenges in the robot cell. To a get better understanding of the requirements for how to include the operator in the process, we are closely cooperating with a number of industrial partners [14].

The work of further developing the configuration tool has been divided into two sub-steps. In the first step we create an adaptable tool where the human activity can be pre-configured and planned, analyzed and implemented during setup of the cell. In the second step this should be possible to do in real-time from the cell while in action, resulting in changed activities for the robot during production. We are currently working on the first step of introducing human activities into the workflow.

A. Adapting the configuration tool

In **Figure 1**, an example of a process flow is illustrated, taken from one of our industrial partners. The flow could be described as a regular UML activity diagram [1], but we want to show both the physical robots and their activities, which is why we use this type of flowchart. The product, illustrated by a triangle, comes into the flow from the left and is then moved from station to station until it has passed through the sequence. Each rectangle in this example represents a machine or robot that performs a specific task. In the example in **Figure 1**, no human activities are included.

If we now introduce activities performed by a human operator, it could look like the flow presented in **Figure 2**. The activity is marked as *"human activity"* during process planning and translated into information to the cell. A formal way to describe the details of the human activity and the interaction could be to use BPEL4People [11] or any other standardized modeling language. By clicking the rectangle marked as a "human activity" a new UML chart or flow chart should be visible showing each step of the task performed by the human. A tree view according to GOMS model (Goals, Operators, Methods, and Selection rules) [4] could be added to the chart to further expand the details. The goal would be the purpose of the human activity – what is the human expected to do. The Operators then describe the particular actions the human must perform to reach the goal. The order of the operators creates a sequence or a Method. There might be more than one viable sequence of actions available and if so the selection rules are used to establish when to use which method. In **Figure 2** the human is expected to perform an inspection and the GOMS documentation should then provide information about how to perform the inspection, what type of inspection it is and if there are alternative ways depending on the product.

From a software-tool perspective, this looks just like a small change but in reality this will have major consequences. The outcome of the human activity has to be communicated to the robots so that the next robot in the flow knows how to act. The communication could be through a broadcast signal or a message sent from the PLC system. The messages also have to be visualized to the operators in the screen. This can be done in several ways and is a matter for future testing.

The overall PLC control system will be responsible for coordinating the work and keep track of the parts. By using a unified process description, the system can keep track of the parts in a sequence of activities.

 ISBN: 978-1-61208-156-4

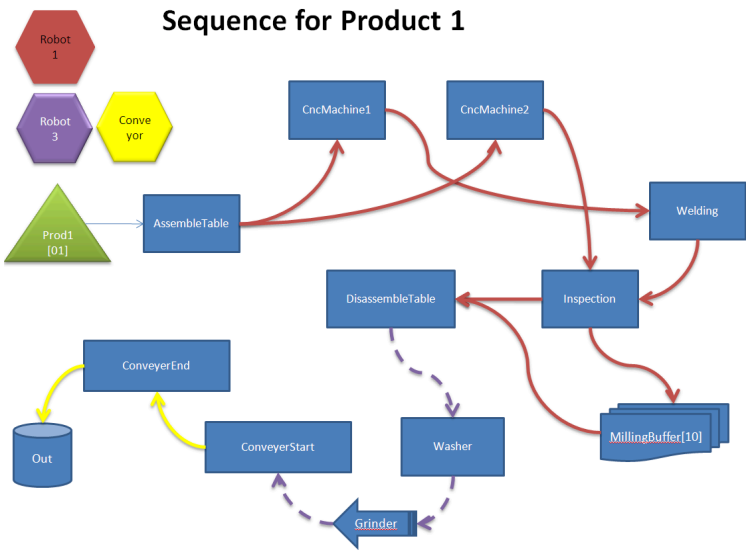


Figure 1. An example of a sequence of activities in a production line, created in our process planning tool.

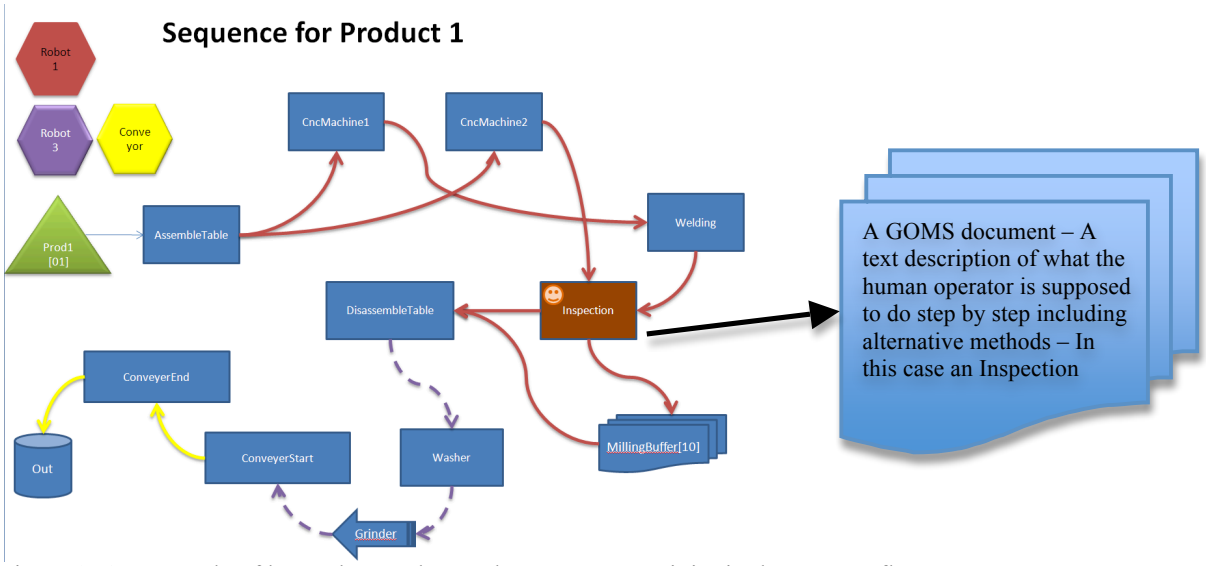


Figure 2. An example of how a human have taken over one activity in the process flow.

B. An interface for comunication between actors

To solve parts of the communication problem we have to turn to the actual robot cell. It is here the interaction is being performed and also here the signals have to be sent from. In **Figure 3** below we see a typical robot cell and a human operator waiting outside for an activity (left side). Next we see how the operator goes into the cell (right side) to perform the tasks. To communicate with the operator it has been concluded that we need a screen of some sort in the cell.

According to our industrial partners the screen should have multiple purposes and communicate different types of information to the operator. The screen should provide the operator with support for the task at hand but also show the overall flow so the operator knows where in the flow they are involved and why.

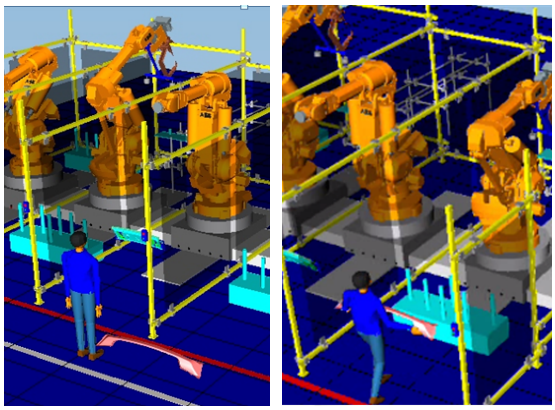


Figure 3. A robot cell where an operator is first waiting for the command to go in (to the left) and then goes into the cell (to the right) to solve a set of tasks.

 ISBN: 978-1-61208-156-4

It is beneficial to use symbols and icons the operators are used to as well as interaction styles that are adapted to the industrial environment [20]. By using an interface with different levels of information the operator can get access to information of the process from a meta level down to detailed information on materials and assemble instructions. The interface should show the process flow so that the operator can see what is going on in the cell in front of him or her. In **Figure 4**, an example of our current design suggestion is illustrated. When an activity in the flowchart is clicked a set of icons in a menu appear and the operator can select what to do by clicking one of the icons. In **Figure 4** the operator has selected the document mode and a group of documents, attached to the activity selected, have appeared in the lower part of the screen. It should then be possible to open up available documentation for that activity and detailed descriptions on the different steps.

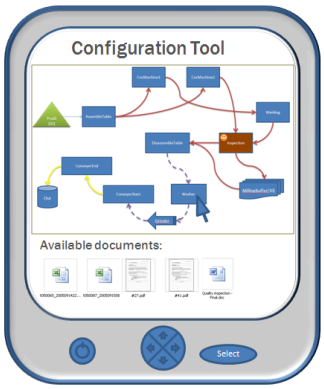


Figure 4. An example of what a screen for cell interaction and process information could look like.

According to our background research, the screen should provide process completion information; time status of the activity. It should also signal to the operator that it is time for human activity within the cell and when to go in (see **Figure 5**). When the operator has entered the cell, the screen should provide information about the tasks at hand, and be able to guide the operator through the tasks by providing text and image descriptions.

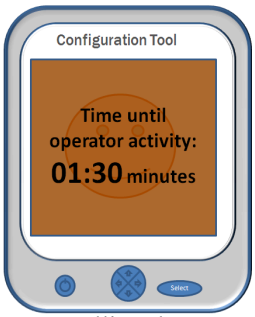


Figure 5. An alert message telling the operator that it is time for him or her to enter the cell in a minute and a half.

In **Figure 5** an example of an alert message is displayed showing that an operator is needed within a minute and a half of time. In this screen we have also concluded that it could be helpful to show instruction for pre-conditions. A pre-condition could be material or a tool needed for the assembly, which the operator must fetch and bring with him or her into the cell before the task can be performed.

C. *Creating a feedback system for interacting actors*

When the robots have performed a task they signal to the PLC that the part can be picked up and moved to the next station. The screens described in previous section should not only function as information displays they should also adapt and provide possibilities for the actors to interact. The screen should be able to signal different scenarios to the operator but also be used to signal feedback to the robot and to the PLC. When an action has been performed or a problem has been solved the human must be able to signal back to the system. **Figure 6** illustrates an example of our current design for what such a screen could look like. In the figure the action Inspection is selected and its work tasks are shown as a list. The list is clickable to provide details for each task. By clicking a task the operator also gets access to forms for measurements, error reports and feedback notes. The operator can use the checkboxes to signal when a task is completed. When all tasks in an action are performed and checked the "completed" button will light up and get available. The operator can then click it and through that signal to the control system that the action is complete and that the part can be moved to the next cell. Depending on the outcome of the activity the PLC will order different actions for the next step in the flow. If for example the part is discovered to be defect during inspection the part should be discard and the PLC will order the moving robots to do this instead of moving it to the next station in the flow.

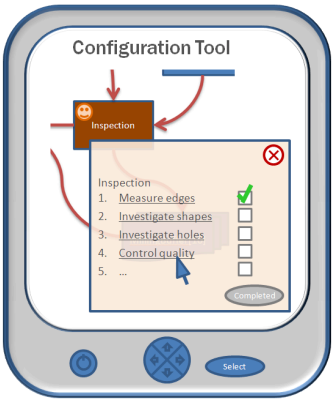


Figure 6. The process flow and instructions for the operator.

IV. DISCUSSION AND CONTINUING WORK

The next step of our work is to gather more information of how to design the interfaces. We will look at icons, click patterns, interactions styles, and colors, as well as contents and information presentation; using established research in the field [20]. The operators will then be involved in usability analyzes [16]. Later on, the human operator should be able to change the process flow during operation by accessing the tool from the cell; changing a *"machine made"*

activity into *"performed by operator"*. According to industry, it would be beneficial if a sensor grid could identify what the operator is doing and by automation change the activities in the process flow and tell the robot how to interact. It should be possible to signal to the system and the robot by hand signals or camera identification when a part for example has been assembled correctly and then update the status of that task in the tool. Research has been done [8, 10, 13, 17-18] on this type of Human-Robot inter-action and should be possible to incorporate in our solution. We have also concluded that the process planning tool has to be able to in real-time analyze a change of procedure and estimate different consequences and produce a risk assessment for the operator. The idea is that the tool should be able to look at the process flow and estimate the consequences of different scenarios. It should be able to tell the operator if it is beneficial to enter the robot cell at that time or if he or she should wait a bit depending on how that interruption affects the robot at that time. The tool should also be able to signal for help when a robot gets overloaded and falls behind the work flow, so that an operator can go in and if possible assist that robot or reassign another robot in the line to help out.

Similar problems have been tackled in research on holonic manufacturing systems [9, 15, 22, 23]. Here each robot, machine or operator is assigned with autonomous and cooperative properties and their activities are determined through cooperation instead of a centralized system. A large proportion of the work done within holonic systems is focused on the technical sides however and our work on including the operator could perhaps add knowledge to that domain. Current work in related projects on artificial intelligence could also render guidance in this matter.

However, this is highly non-trivial and it will require solutions both from our project and parallel projects within FLEXA [18, 19] and Flex Lean [6, 7] as well as related work within the research community. The work from several projects will eventually be turned into one or two applications that will be used within our partner industries.

ACKNOWLEDGEMENT

The authors would like to thank Eric Lind for editorial help and our cooperating partners for sponsoring our project.

REFERENCES

[1] S. W. Ambler 2004, The object primer : agile model-driven development with UML 2.0, 3. ed. Cambridge: Cambridge Univ. Press, UK

[2] K. Behnisch, "Taming the robot - Better safety without higher fences", in Robotics Highlights ABB Review 4, 2006

[3] K. Bengtsson et al. "Relations identification and visualization for sequence planning and automation design", in Proc of 2010 IEEE Conference on Automation Science and Engineering (CASE), 21-24 Aug. 2010, p. 841.

[4] S. Card, T.P. Moran and A. Newell (1983). The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates.

[5] F. Danielsson, "A Flexible Process Planning Platform for Flexible Automation", Project presentation, University West, 2011, [Last access: 2011-05-14], [Electronic] http://www.cdti.es/recursos/doc/eventosCDTI/Aerodays2011/1D4.pdf

[6] F. Danielsson, B. Svensson, and S Gustavsson, "A Flexible Lean Automation Concept for Robotized Manufacturing Industry", In Proc. of 11th Middle Eastern Simulation Multiconference: Alexandria, Egypt, Dec. 2010, pp. 101-104.

[7] FLEXA - Advanced Flexible Automation Cell, A reseach project within the European Seventh Framework Programme (FP7). [Electronic:] http://www.flexa-fp7.eu, [Last acess: 2010-05-28].

[8] M. A. Goodrich and A. C. Schultz,"Human-robot interaction: a survey", In Foundations and Trends in Human-Computer Interaction Volume 1 Issue 3, February 2007.

[9] L. Gou, P.B. Luh and Yuji Kyoya, "Holonic manufacturing scheduling: architecture, cooperation mechanism, and implementation", in Computers in Industry 37, Elsevier, 1998, pp. 213-231,

[10] S.T. Hayes, E.R. Hooten, and J.A. Adams, "Multi-touch interaction for tasking robots", In Proc. of the 5th ACM/IEEE international conference on Human-robot interaction (HRI '10), Osaka, Japan, Marsh, 2010, pp. 97-98.

[11] T. Holmes, H. Tran, U. Zdun, and Schahram Dustdar, "Modeling Human Aspects of Business Processes – A View-Based, Model-Driven Approach". in Proc. of the 4th European conference on Model Driven Architecture: Foundations and Applications (ECMDA-FA '08), Berlin, Germany, June, 2008.

[12] S. Kock, J. Bredahl, P.J. Eriksson, M. Myhr, and K. Behnisch, "Safe collaboration with ABB robots - Electronic Position Switch and SafeMove", White Paper ABB Robotics AB, 2008

[13] K. Nickel and R. Stiefelhagen, "Visual recognition of pointing gestures for human–robot interaction", In Elsevier, Image and Vision Computing Volume 25, Issue 12, 3, Dec. 2007, pp. 1875-1884.

[14] The Production Technology Centre, [Electronic] http://www.innovatum.se/pages/ptc_uk_partners-3121.html [Last access: 2011-05-14]

[15] A. Rosu, "PLC-based Holonic manufacturing cell transport system", U.P.B. Scientific Bulletiene, Series C, Vol. 73, Issue 2, 2011, pp. 117-126.

[16] J. Rubin, and D. Chisnell, (2008), Handbook of Usability Testing - How to Plan, Design, and Conduct Effective Tests, 2nd Ed. WILEY publishing Inc. Indianapolis, USA.

[17] E. Sato, T. Yamaguchi, and F. Harashima, "Natural Interface Using Pointing Behavior for Human–Robot Gestural Interaction", In IEEE Transactions on Industrial Electronics, Volume 54, Issue 2, April 2007, pp. 1105-1112.

[18] R. Stiefelhagen, et al., "Natural human-robot interaction using speech, head pose and gestures," in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., 2004. pp. 2422–2427.

[19] N. Sundström, "Automatic Generation of Operations for the FLEXA Production System", Master of Science Thesis in Automation, Depertment of signals and systems, Chalmers University of Technology, Göteborg, Sweden, 2010.

[20] B. Shneiderman and C Plaisant (1998) Designing the User Interface-Strategies for Effective Human-Computer Interaction. 4th ed. Addison Wesley Longman, Inc. USA.

[21] D.J. Smith (2010) The Safety Critical Systems Handbook, Butterworth-Heinemann, Oxford, UK.

[22] H. Sun and P.K. Venuvinod, "The human side of holonic manufacturing systems", in Technovation, Volume 21, Issue 6, June 2001, pp. 353-360

[23] J. Wyns, Concepts for Holonic Manufacturing, Katholieke Universiteit, Leuven, 1996, [Electronic]: http://www.mech.kuleuven.ac.be/pma/project/goa/concepts.htm [Last access: 2011-05-14]

 ISBN: 978-1-61208-156-4

Fungi as Metaphors for Resource Management

Eilidh McAdam, Ruth E Falconer and James Bown
University of Abertay
Dundee, UK
Email: e.mcadam@abertay.ac.uk, r.falconer@abertay.ac.uk, j.bown@abertay.ac.uk

John Crawford
University of Sydney
Sydney, Australia
Email: john.crawford@sydney.edu.au

***Abstract*—Modern societies are heavily dependent upon a number of critical infrastructure networks that allow our societies to function, including water, power and transportation. These networks are open to failure through a range of processes including shortage of essential resources, breakdowns at key nodes and surges in demand and this means that effective management of such networks is challenging. The fungi, an entire kingdom of life, epitomise successful networks in nature, demonstrating pervasive growth in harsh environments and a capacity to optimise resource distribution in space over time. We detail a graph-based implementation of an established model of fungal colony growth and recycling that provides a scheme for efficient resource allocation and distribution. Local processes of uptake of resource, growth and biomass recycling lead to non-local emergent patterns of biomass distribution and colony functioning. This colony growth and recycling in response to changes in resource supply leads to highly flexible spatio-temporal resource balancing in the natural world. In our simulations, we identify fungal colonies that have genotypes that maximise throughput from prescribed sources to sinks in a spatial environment, and provide a biological interpretation to reveal the link between the observed macro-scale pattern and the micro-scale processes that effect a given solution. This interpretation is then contextualised in the domain of critical infrastructure management.**

***Keywords*-adaptive systems; critical infrastructure; biological metaphors.**

I. INTRODUCTION

Modern societies are heavily dependent upon a number of critical infrastructures, including energy, transport, water and telecommunications, that allow our societies to function in an effective manner [1]. These infrastructures are interconnected and often have common resource dependencies [2], and have often evolved over long periods of time without foresight of the extent of future demands or system interconnectivity [3]. Rinaldi et al. characterise this interconnected, interdependent mix of networks as a complex adaptive system [4] where the behaviour of the whole system is not describable in terms of the component parts alone but in terms of emergent properties of the whole system. Ulieru highlights that to preserve functioning across this adaptive system, a management system must be able to dynamically reconfigure itself through self-organisation and self-repairing processes to effect resilience to perturbation [5], and in later work suggests that natural systems may offer routes to solutions [6] since natural systems have many of the properties sought after in complex adaptive systems management.

The use of biological metaphors to inspire algorithms for solving complex problems is already a well developed field with a long history [7]. Ant colonies have been used as a metaphor for routing algorithms in telecommunications networks [8]. Recent work in this field has extended the metaphor to consider ad-hoc networks [9] and trust provision in wireless networks [10]. Other biological metaphors for network management include bees [11] and cell biology [12]. In the field of biological metaphors, a little explored but highly successful organism that exhibits features that are desirable for protecting societys critical infrastructures are fungi.

Fungi are an entire kingdom of life and are one of the most successful organisms on the planet, with estimates suggesting that there may be as many as 1.5 million species of fungi globally [13]. A fungal colony is a highly successful organism, demonstrating pervasive growth through and survival in harsh environments such as soil. They achieve this through their capacity to operate in a decentralised manner, reacting locally to changes in context while interoperating at the colony scale and with other organisms and the environment [14].

Fungi are decentralised transport networks of hyphae that grow and branch at micrometre scales through the tortuous porous structure of soils [15]. Resource is taken up locally through the tips of the hyphae and then redistributed globally through translocation within the colony [16]. Colonies can recycle biomass: resource that is converted into hyphal structure may be remobilised through degradation of local regions of hyphae and translocated elsewhere in the colony [17]. This provides fungi with a highly plastic phenotype, where biomass investment may be dynamically linked to external resource availability, and local behaviour impacts on and is impacted by non-local behaviour in the colony resulting in emergent phenotypes [14].

Bebber et al. recognise that understanding how decentralised, highly flexible fungal colonies that are capable of pervasive growth, emergent behaviour and self-healing may inform the design of man-made networks [18]. They show

 ISBN: 978-1-61208-156-4

that fungal colonies are effective transport networks that allocate limited resource efficiently. However, models of fungal colonies from which algorithms may be developed are scarce [19].

Like fungi, critical infrastructures can be modelled as graphs [20], i.e. a set of vertices and edges that represent the critical infrastructure components and connections. Graphs provide a proven way of representing critical infrastructures under a wide range of configurations and for many different domains [2], [20], [21].

Given the superior resource management strategies fungi adopt we expore the fungi as a metaphor for efficient resource management in critical infrastructure networks. Here, we examine the trait set that could lead to the development of a novel bio-inspired algorithm based on [17] for resource allocation in network structures, exploiting key properties of fungal colonies to establish a decentralised resource management. Consequently, an important first step is to generalise the formulation of the computational model from the existing 3D lattice implementation to a graph-based implementation. We show that modelled graph-based fungal colonies still exhibit the essential emergent phenomena that make fungal colonies such attractive metaphors.

In our first simulations, we identify the genotypes of fungal colonies that perform well with regards to network throughput and provide a biological interpretation to reveal the link between the observed performance and the model parameters that effect a given result. This interpretation is then contextualised in the domain of critical infrastructure management.

II. Methodology

As a fundamental step in understanding the origins of the fungal phenotype we developed a model of fungal colony dynamics and interactions that includes an explicit account of the physiology of organisms in order to link properties of the emergent phenotype to underlying physiological processes [17]. The model formulation represents individual mycelial networks growing in the environment as comprising three fractions: insulated biomass, non-insulated biomass and mobile biomass. These essentially relate to but are not limited to older inactive biomass, active hyphal tips and internal resource respectively. The relative proportion of these components is dynamic and determined by four physiological processes: uptake, inter-conversion between mobile and insulated/non-insulated phases (recycling of biomass), redistribution of mobile biomass and growth. Further, a fungal individual is characterised by a trait set (genotype) which regulates the physiological processes and its interaction with the environment. The model is based on a set of partial differential equations (Figure 1) which represent the interdependencies amongst the types of biomass - non insulated (b_n), insulated (b_i) and mobile biomass (n) and

$$\frac{\partial b_i}{\partial t} = \zeta[\Delta b_n + \beta_2(\alpha_n \pi^\theta - \beta_n \pi)] + \beta_2(\alpha_i \pi^\theta - \beta_i \pi) b_i$$

$$\frac{\partial b_n}{\partial t} = (1-\zeta)[\Delta b_n + \beta_2(\alpha_n \pi^\theta - \beta_n \pi) b_n]$$

$$\frac{\partial n}{\partial t} = \Delta b_n - \beta_1((\alpha_n \pi^\theta - \beta_n \pi) b_{ni} + (\alpha_i \pi^\theta - \beta_i \pi) b_i) + (\lambda_1 b_n + \lambda_2 b_i) s$$

$$\frac{\partial s}{\partial t} = \omega(s_{max} - s) - (\lambda_1 b_{ni} + \lambda_2 b_i) s$$

Figure 1. Mathematical model describing colony growth and resource management based on [17]

external resource (s) - and how these change over space and time [17].

The continuous model outlined above was discretised and solved using a square lattice. In order to obtain a discrete graph based approach to fungal colony growth space is now represented as a hexavalent lattice. The colony is represented by a set of edges constrained by the architecture of the hexavalent lattice and each edge may contain all three fractions of biomass: insulated, non-insulated and mobile as detailed above. The edges can also possess a prescribed level of external resources. The key biological processes described by Falconer's model [17] are also implemented over the graph based colony.

Here, simulation of the dynamics of a single colony is performed and measurement is carried out using resource sources and sinks. Sources of resource, loosely simulating patches of concentrated organic material, can be placed in set locations in the colony allowing the uptake process to take place and mobile biomass to be introduced to the colony. Resource sinks represent areas of the colony investing in, for example, fruiting bodies or colony defenses and remove mobile biomass at a constant rate. Strategically placed sources and sinks are used to measure the throughput and response of the colony.

The complex interplay between the trait values and the size of the search space led to a genetic algorithm being used to find the genotypes which performed best. During the simulation, external resource was introduced into colony through three sources. Three sinks on the opposite side of the colony were set up to remove all mobile biomass that diffused into them. Because the amount of resource going into the colony was constant, the fitness used was simply the total amount of biomass that came through the sinks. Table I lists the traits that were being altered and the meaning of each. Selection was performed using linear rank selection and stochastic universal sampling [22].

A population of 40 individual colonies were used in each generation of the genetic algorithm. With regards to simulation time, two scenarios were examined: 100 and

 ISBN: 978-1-61208-156-4

Trait	Description
α_i	Insulated immobilisation [0-1]
α_n	Non-insulated immobilisation [0-1]
β_i	Insulated mobilisation [0-1]
β_n	Non-insulated immobilisation [0-1]
ζ	Insulation of non-insulated biomass [0-1]
θ	Non-linear term [1-3]

Table I
ANNOTATED LIST OF TRAITS (WITH SEARCHABLE RANGES) FROM THE COMPUTATIONAL MODEL OF [17] WITH RESPECT TO THE EQUATION SET IN FIGURE 1

300 time steps. These values were derived from preliminary studies (not shown). The initial run of 100 time steps resulted in individuals that had a peak of sink activity just before simulation termination. To determine whether other trait sets would outdo the fittest in this scenario if given more time to become established, the genetic algorithm was also run with the simulation for each individual being run for 300 time steps.

III. RESULTS

Falconer et al. (2005) reproduced observable growth patterns in fungal colonies and demonstrated plausible emergent behaviour. Qualitative comparison of fungal phenotypes using the same parameters as in Falconer et al., 2005 also showed good correspondence between the continuous and graph-based implementations (figure 2). In particular, we looked for the continuous model's "fairy ring" pattern and concentric rings (Figure 2) as well as a plateau biomass profile. These patterns were also observable in the discretised graph-based model.

In both scenarios, the average fitness in a generation of the genetic algorithm went up over time. While we tried a run using 200 generations, we found that the average fitness plateaued before reaching 100 generations and so the results from this are not included. While the actual values of the traits of the fittest individuals differ, when the ratio of α_i/α_n and β_i/β_n (insulated/non-insulated immobilisation and mobilisation traits) are viewed by fitness, a clear separation is found. Figure 3 shows each individual's immobilisation and mobilisation ratios against their fitness and shows that fit individuals tend to have an immobilisation ratio of between 0 and 1 and a mobilisation ratio between 1 and 2. No strong trends were found involving the two remaining traits, ζ and θ.

IV. DISCUSSION

Different colony foraging strategies, i.e. the way in which the colony extends itself in space over time and the manner in which the colony growth pattern responds to resource acquisition - are reflected in the physiological trait set. Consequently, different trait sets may be defined to determine different resource management strategies - manifest through dynamic patterns in growth - in critical infrastructures.

We demonstrate the use of a genetic algorithm search to identify colony trait sets that provide optimised solutions to a specific resource management task. Here, in our initial exploration to determine the most efficient trait sets associated with maximising throughput with constant source-sink relationships the recycling trait ratios (α_i/α_n, β_i/β_n) converged to [0 -1] and [1-2] respectively across the multiple runs of the GA.

For this scenario we may interpret this as the fastest investment (in terms of spatial extent) into a fixed and evenly distributed infrastructure. This even distribution occurs via the reallocation of resources from places where it already exists (recycling of insulated structural biomass, i.e. $\alpha_n > \alpha_i$) to areas where it is required to be established (periphery of colony, i.e. $\alpha_n > \alpha_i$). This localised, dynamic recycling behaviour thus maximizing throughput at the global level.

We expect this relation to be different, or at least the magnitude of the ratios to be different, for maximum throughput associated with changing source-sink relationships and/or damage to the network. In these contexts the resource allocation and distribution mechanism must be adaptive in space over time. We propose that our framework provides a platform to explore how the colony traits may interact to promote plasticity in form. This plasticity can allow rapid switches in foraging strategy and we suggest that a fungal colony inspired resource management approach may offer potential in protecting societys critical infrastructures.

V. CONCLUSIONS AND FUTURE WORK

The results reported here consider only a static environment and the capacity of a dynamic colony structure to optimise resource throughput. Clearly, a static solution may be identified a priori to such a problem. The most compelling motivation for using fungi as a metaphor for resource management in critical infrastructures is the capacity of colonies to react dynamically to a changing resource regime.

Importantly, our modelling framework does indeed allow configuration of dynamic environments and simulation of colony behaviour in that dynamic environment. We will use the same discretised colony growth and recylcing model coupled to the genetic algorithm search approach to optimise traits for dynamic, challenging problems. In particular, we will consider two main classes of problems. First, we will identify colonies that are able to deal well with the challenges of load balancing. To explore this we must vary over time the level of resource at the sources and measure, as here, the resource at the sinks. The genetic algorithm search may optimise throughput in the same way as detailed here.

Second, and more interestingly, we will identify colonies that are resilient to damage. Damage will be effected by removing sections of the network and again optimising on throughput. In both cases, we will undertake systematic experimentation and seek biological interpretations of the resulting optimised trait sets.

 ISBN: 978-1-61208-156-4

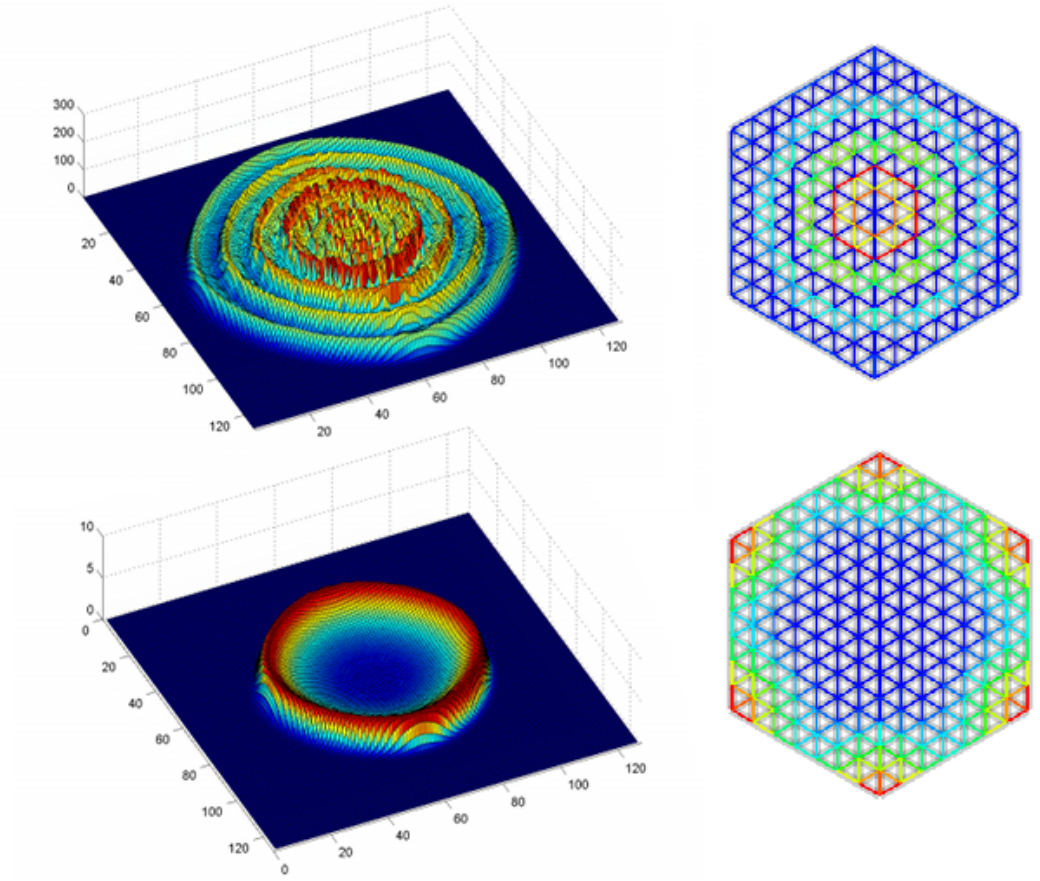


Figure 2. Comparison between continuous model (left) and graph based model (right).

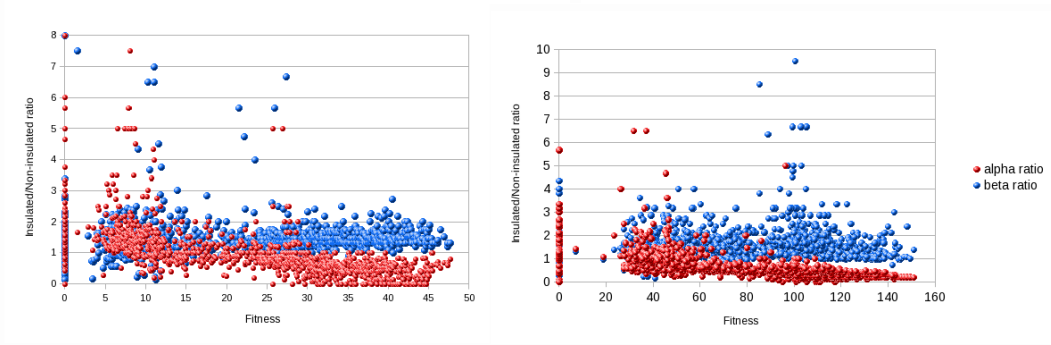


Figure 3. Trait ratios - 100 time step (left) and 300 time step (right) scenarios

It is anticipated that these analyses will reveal fungi to be a useful, versatile scheme upon which to base management algorithms in network-based contexts characterised by uncertainty and variance in demand loading and by the requirement to function in the face of partial failure.

REFERENCES

[1] E. M.-K. A. Boin, P. Lagadec and W. Overdijk, "Critical infrastructures under threat: Learning from the anthrax scare," *Journal of Contingencies and Crisis Management*, vol. 11, no. 3, pp. 99–104, 2003.

[2] N. K. Svendsena and S. D. Wolthusen, "Connectivity models of interdependency in mixed-type critical infrastructure networks," *Information Security Technical Report*, vol. 12, no. 1, pp. 44–55, 2007.

[3] M. Amin, *Automation, Control and Complexity: An Integrated*

 ISBN: 978-1-61208-156-4

Approach. Hoboken, NJ: John Wiley and Sons, 2000, pp. 263–286.

[4] J. P. S. Rinaldi and T. Kelly, "Identifying, understanding and analyzing critical infrastructure interdependencies," *Control Systems, IEEE*, vol. 21, no. 6, pp. 11–25, 2001.

[5] M. Ulieru, "e-networks in an increasingly volatile world: Design for resilience of networked critical infrastructures," in *Proc. The Inaugural IEEE International Conference on Digital Ecosystems and Technologies*, Cairns, Australia, Feb. 2007, pp. 540–545.

[6] M. U. V. Vyatkin, G. Zabelova and D. McComas, "Toward digital ecologies: Intelligent agent networks controlling interdependent infrastructures," in *Proc. 1st IEEE Conference on Smart Grid Communications*, Gaithersburg, MD, USA, Oct. 2010, pp. 589–594.

[7] J. d. L. J. Martin and D. Maravall, "Adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature," *Natural Computing*, 2009.

[8] M. B. M. Dorigo and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.

[9] F. D. G. di Caro and L. Gambardella, "Anthocnet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks," *European Transactions on Telecommunications*, vol. 16, no. 5, pp. 443–455, 2005.

[10] G. P. F. Marmol, "Security threats scenarios in trust and reputation models for distributed systems," *Computers & Security*, vol. 28, no. 7, pp. 545–556, 2009.

[11] S. K. M. Saleem and M. Faroog, "A formal performance modeling framework for bio-inspired ad hoc routing protocols," in *Proc. 10th annual conference on genetic and evolutionary computation*, Atlanta, Georgia, USA, Jul. 2008, pp. 103–110.

[12] F. Dressler, "Efficient and scalable communication in autonomous networking using bio-inspired mechanisms - an overview," *Informatica*, vol. 29, no. 2, pp. 183–188, 2005.

[13] D. Hawksworth, "The fungal dimension of biodiversity: magnitude, significance, and conservation," *Mycological Research*, vol. 95, no. 6, pp. 641–655, 1991.

[14] J. B. et al., "Evidence for emergent behaviour in the community-scale dynamics of a fungal microcosm," *Proceedings of the Royal Society B*, vol. 266, no. 1432, pp. 1947–1952, 1999.

[15] K. Ritz and I. M. Young, "Interactions between soil structure and fungi," *Mycologist*, vol. 18, no. 2, pp. 52–59, 2004.

[16] A. Ashford and W. Allaway, "The role of the motile tubular vacuole system in mycorrhizal fungi," *Plant Soil*, vol. 244, no. 1-2, pp. 165–175, 2002.

[17] N. W. R.E. Falconer, J.L. Bown and J. Crawford, "Biomass recycling and the origin of phenotype in fungal mycelia," *Proceedings of the Royal Society B*, vol. 272, no. 1573, pp. 1727–1734, 2005.

[18] D. Bebber, J. Hynes, P. Darrah, L. Boddy, and M. Fricker, "Biological solutions to transport network design," *Proceedings of the Royal Society B*, vol. 274, no. 1623, pp. 2307–2315, 2007.

[19] T. R. A. Schnepf and P. Schweiger, "Growth model for arbuscular mycorrhizal fungi," *Journal of the Royal Society Interface*, vol. 5, no. 24, pp. 773–784, 2007.

[20] V. Latora and M. Marchiori, "How the science of complex networks can help developing strategies against terrorism," *Chaos, Solitons & Fractals*, vol. 20, no. 1, pp. 69–75, 2004.

[21] A. Krings and A. Azadmanesh, "A graph based model for survivability applications," *European Journal of Operational Research*, vol. 164, no. 3, pp. 680–689, 2005.

[22] F. Herrera, M. Lozano, and J. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 187–204, 2000.

 ISBN: 978-1-61208-156-4

Adapting to the Unknown With a few Simple Rules: The glideinWMS Experience

Igor Sfiligoi, Benjamin Hass, Frank Würthwein
University of California San Diego
La Jolla, CA 92093, USA
email: {isfiligoi,bhass,fkw}@ucsd.edu

Burt Holzman
Fermi National Accelerator Laboratory
Batavia, IL 60510, USA
email: burt@fnal.gov

***Abstract*—The High Energy Physics community requires a large amount of compute resources and had to adopt the Grid computing paradigm to satisfy its needs. Scheduling of jobs in the Grid environment is however very challenging, due to the high autonomy enjoyed by the participating resource providers, requiring the user community to constantly adapt to the ever-changing conditions. The CMS experiment addressed this problem by developing the glideinWMS system, which addresses this problem by using an overlay compute pool and a few simple rules for provisioning the needed resources. This approach has been very successful and is now being used by several other communities in the Open Science Grid. This paper provides the description of the glideinWMS system, the algorithms used as well as an analysis of the experience CMS has had using the system.**

Keywords-layered scheduling; adaptation; glideinWMS.

I. INTRODUCTION

Over the past decade, the high throughput computing in science has been moving from dedicated compute clusters to a widely distributed, shared Grid infrastructure in an effort to distribute the system maintenance effort, increase the average equipment utilization and gather additional compute resources in times of need. This paper explores the challenges of doing Grid-wide user job scheduling in this environment.

One of the core principles of the Grid paradigm that makes it so appealing for the scientific resource providers is the high autonomy enjoyed by each participating compute cluster, allowing them to participate in the system without sacrificing neither the quantity nor the quality of compute resources given to the local users. As a consequence, while the external, opportunistic users can request to use compute resources, they have no guarantee if and when those compute resources will become available.

Another core principle that makes the Grid paradigm appealing to the user community is, instead, the freedom users retain to schedule the compute resources the way they deem more fit; the users are free to choose the product they like, or for example, only submit to local resources due to the perceived ease of use. As a consequence, there cannot be a single Grid-wide job scheduler instance, and there is also no guarantee that the various instances will exchange information with one another.

A Grid-wide scheduler is thus unlikely to ever obtain an accurate state of the whole system, much less be able to predict what the state of the system will be in the near future. Being able to adapt to the ever-changing situation is thus essential.

This paper describes the approach taken by the glideinWMS system [1,2], a scheduling solution developed by the Compact Muon Solenoid (CMS) experiment [3] and extensively used in the Open Science Grid (OSG) [4,5]. The key component of this system is the conceptual simplicity of the approach; the user scheduling is solved by the use of an overlay compute pool and the resource provisioning is handled by just a few simple rules.

A schematic description of the system architecture is provided in Section II, while Section III provides a detailed description of the resource provisioning rules. Section IV provides an analysis of the experience CMS has had using the system. Finally, Section V provides a comparison against other Grid-wide scheduling systems.

II. THE GLIDEINWMS ARCHITECTURE

The glideinWMS approach to Grid-wide user job scheduling is based on the pilot paradigm. In this paradigm, the scheduling system does not even try to directly schedule the user jobs on Grid resources, but instead creates a dynamic overlay pool of compute resources on top of the Grid resources, by submitting so-called **pilot jobs**, and then schedules user jobs inside this pool. A schematic view of a pilot system is shown in Fig. 1. More details can be found in [1].

The pilot jobs are effectively resource provisioners; once one of them starts on a Grid resource, it takes ownership of that resource for the allocated lease time, and gives it in exclusive use of the pilot system, by joining the overlay pool. The pilot system scheduler thus has complete control over this overlay pool, and can make scheduling decisions based on trustworthy information obtained from the provisioned resources, just as in a truly dedicated compute cluster.

The pilot system, of course, now needs to schedule the pilots themselves across all the Grid resources, and do this with only partial information. Due, however, to the nature of pilot jobs, this task is much simpler than direct Grid-wise user job scheduling. Unlike user jobs, all pilot job payloads are the same, and thus the order in which they start is irrelevant. Moreover, each and every pilot can run jobs from any user of the community, relinquishing the need of inter-

 ISBN: 978-1-61208-156-4

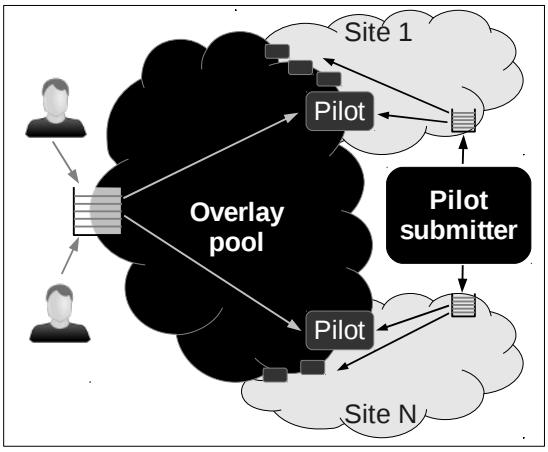


Figure 1. A pilot system

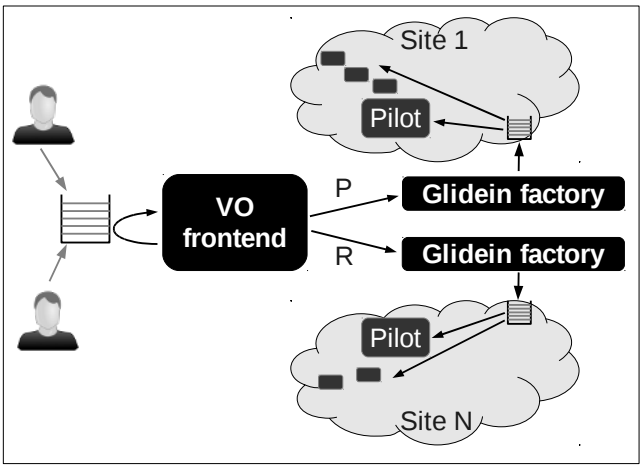


Figure 2. The glideinWMS pilot submission

pilot priorities. The glideinWMS scheduling algorithms exploit these properties and do not track the single pilot jobs themselves, but only monitor and regulate the cardinality of pilot jobs in the queues.

Somebody, however, still has to actually submit and track the single pilot jobs as they are submitted to the Grid sites. In the glideinWMS system this is delegated to a set of processes called **glidein factories**, one per logical Grid site, which are essentially just slaves to the actual scheduling process, called the **VO frontend**, as shown in Fig. 2.

The communication between the frontend and a factory is based on the concept of **constant pressure**; the frontend asks the factory to keep a certain number of pending, or idle pilot jobs in the Grid queue, and to continue to submit new ones to replace the ones that start running, until the frontend issues a new request changing that number, possibly to zero. In Fig. 2, the pressure numbers are represented by letters P and R.

The adaptability of the system thus lies in the calculating, at any given point in time, the appropriate pressure point for each and every Grid site. If the pressure is too low, there may not be enough idle pilots in a Grid site's queue when compute resources at the site become available, resulting in a smaller overlay pool and thus lower user job throughput. If instead the pressure is too high, the Grid resources added to the overlay pool may not be needed anymore by any user job, resulting in wasted CPU cycles.

III. Regulating the Pilot Pressure

The pilot jobs are being submitted to Grid sites because there is an expectation that when they do start up and provide compute resources to the overlay pool, there will be user jobs that can make use of them. However, in order to accurately forecast the available jobs at pilot startup, the system would need to know the current state of the users' job queue, the Grid site scheduling policies, the behavior of all other Grid-wide schedulers, the behavior of the local users and the run times of the users' jobs. Only the first one is available to the glideinWMS VO frontend.

Most of the logic is thus based on the current state of the users' job queue. At each iteration, the VO frontend collects the information about all the idle user jobs and matches them against the information available about the Grid sites, similarly to how the matchmaker in the overlay pool would do it. The details of what information is available and how is the matchmaking performed is beyond the scope of this document, and the interested reader should refer to the glideinWMS manual [6] instead.

Simply calculating the number of idle user jobs for every Grid site is however not enough. If a user job is capable of running on more than one Grid site, simply counting each job against each matching site will result in double counting for some. The VO frontend thus keeps track of how many Grid sites each job matches against, and counts each job as only the appropriate fraction against each matched site. As an example, if a job matches against sites A, B and C, it will be counted as 1/3 against each of them.

Once the weighted count of matched idle user jobs is computed, the VO frontend calculates the pressure point for each Grid site as a function of the number of matched idle user jobs, as in

$$P_s(t)=f(I_s(t)). \quad (1)$$

As stated above, knowing the current state of the users' job queue is not enough to obtain the optimal value. However, given that there is no easy way to obtain the vast majority of data needed for a reliable forecast, the VO frontend does not even try. Instead, the VO frontend uses a simple heuristic to achieve the desired result.

In our multi-year experience of using the Open Science Grid, we noticed that Grid jobs tend to start and terminate with a relatively flat frequency. Most Grid sites will start and terminate O(10) jobs every few minutes, and it is very rare to have O(100) Grid jobs terminate in the same period. As such, having a pressure point of O(10) is sufficient to get access to the vast majority of available Grid resources.

With the maximum pressure point capped at O(10), the remaining range is small enough to not require fine tuning. The VO frontend thus simply divides the number of currently idle user jobs by 3, resulting in the following formula:

$$f(I_s(t)) = \lceil min(I_s(t)/3, C_s) \rceil. \quad (2)$$

 ISBN: 978-1-61208-156-4

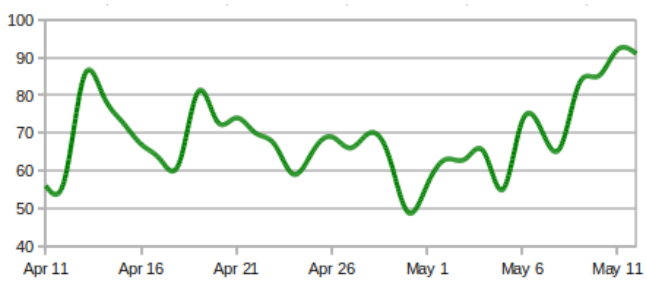


Figure 3. Number of Grid sites used by the CMS glideinWMS system

The factor 3 was chosen pretty arbitrarily, but following the simple logic that the pressure point should be lower than the number of idle jobs, in order to not over-provision, and that the fraction should still be high enough to obtain the desired amount of Grid resources at an acceptable rate.

IV. Operational experience

The CMS experiment has been using the glideinWMS system for over two years and has been generally very satisfied with the experience. The glideinWMS instance at UCSD is serving a user community of about 4k users and scheduling their jobs on Grid resources distributed across about 100 Grid sites located in the Americas, Europe and Asia. The actual numbers of Grid sites used by user jobs in a recent month can be seen in Fig. 3.

As simple as the algorithms described above are, the system proved to be very effective and efficient. Fig. 4 contains the status of the users' job queue in a recent month, both idle and running. As can be seen, the CMS users have been using up to about 16k CPUs in that period, with steep ramp-ups and ramp-downs based on user jobs demand. This resulted in short wait times for user jobs; as shown in Fig. 5, most jobs started within an hour.

Furthermore, as an indicator of the overall system efficiency, the amount of over-provisioned resources, labeled as *Unmatched* in both Fig. 4 and Fig. 6, has been consistently very low and typically represented less than 5% of all the provisioned resources.

The glideinWMS systems has also been recently adopted by several other OSG communities [7], with similar results. It is worth noting that the addition of several other independent glideinWMS frontend instances has had no impact on the performance of the CMS frontend.

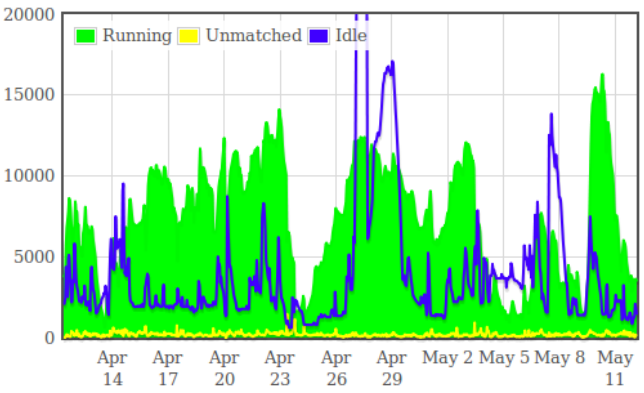


Figure 4. Snapshot of the CMS glideinWMS system

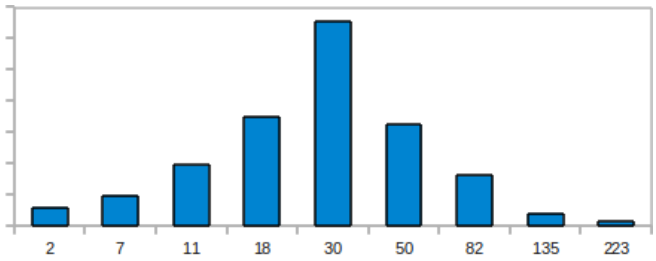


Figure 5. CMS user jobs wait times distribution, in minutes

V. Related work

There are several other products that provide Grid-wide scheduling. They can be categorized as being either direct-submission or pilot systems.

Two major direct-submission systems are the gLite Workload Management System (gLiteWMS) [8] and the Resource Selection Service (ReSS) [9] based OSG Matchmaker (OSGMM) [10]. Compared to glideinWMS, both require a more complex setup by requiring continuous information flow from each and every Grid site. This approach also is less flexible and more brittle, since the information source is controlled by the site, and thus cannot be influenced or verified by the scheduling system; this is not a problem for pilot-based systems like glideinWMS, because the needed information is collected directly by the pilots themselves.

Major pilot-based systems are PanDA [11], DIRAC [12] and MyCluster [13].

The PanDA approach to scheduling of pilots to Grid sites is even simpler than the glideinWMS approach; the system continuously submits pilots to all available Grid sites. If there are no suitable user jobs available at pilot startup, it quickly exists, wasting very little wallclock time. The major drawback of this approach is the high load it imposes on the batch system of every Grid site. By contrast, the glideinWMS system only sends out pilots that are expected to be needed by user jobs, although it too will quickly terminate them if the submission logic was faulty and there are no suitable user jobs available.

DIRAC and MyCluster rely instead on separate services running at each Grid site. This allows them to gather detailed information about the Grid sites, and thus making a more informed decision. The major drawback of this approach is

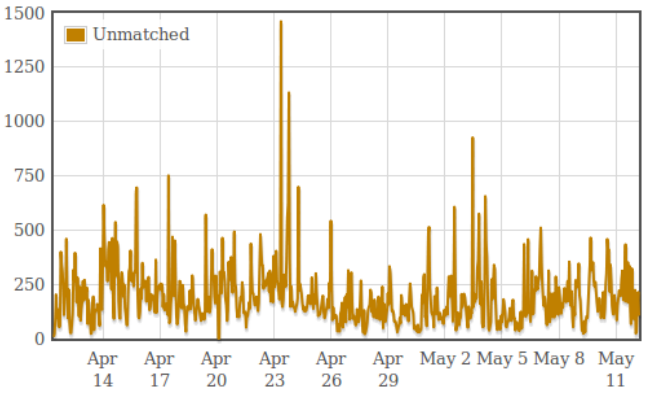


Figure 6. Over-provisioned resources in the CMS glideinWMS system

 ISBN: 978-1-61208-156-4

the assumption that Grid sites will allow the Grid schedulers to install long-lived services at the sites; while this is possible at some sites, many others do not allow this option, thus severely limiting the amount of resources that can be gathered using those systems.

There are also several pilot-based scheduling systems that require the users to explicitly request pilots at various Grid sites. These manually-provisioned systems thus lack end-to-end automation, and a comparison to the glideinWMS would not be meaningful.

VI. Conclusions

The Grid computing environment has many advantages for compute resource providers, but does introduce significant challenges for effective user job scheduling. The main issue is the lack of complete information, which requires continuous adaptation of a Grid-wide scheduling system of each and every user community.

The glideinWMS system addresses this problem by reducing the scheduling complexity through the adoption of the pilot paradigm, where only the pilot jobs need to be scheduled across Grid sites. The user jobs are instead handled within the resulting well-behaved overlay compute pool.

Unlike user jobs, all pilot jobs carry the same payload. Together with the fact that there is only one pilot user, this allows the glideinWMS system to only consider the cardinality of the pilot jobs, drastically reducing the scheduling complexity.

Moreover, operational experience tells us that the Grid job startup frequency is very low, allowing for the capping of the pilot job pressure at an equally low number without significant loss in effectiveness. This makes scheduling of pilot jobs effectively trivial. And the years of operational experience CMS has with the glideinWMS system confirm that this approach works very well.

Using the pilot paradigm and looking only at the cardinality of the pilot jobs thus allow us to reduce the hard Grid-wide scheduling problem into a mostly trivial endeavor.

Acknowledgment

This work is partially sponsored by the US Department of Energy under Grant No. DE-FC02-06ER41436 subcontract No. 647F290 (OSG), and the US National Science Foundation under Grant No. PHY-0612805 (CMS Maintenance & Operations).

References

[1] I. Sfiligoi et al., "The pilot way to grid resources using glideinWMS," CSIE, WRI World Cong. on, vol. 2, pp. 428-432, 2009, doi: 10.1109/CSIE.2009.950.

[2] "glideinWMS," http://tinyurl.com/glideinWMS, Accessed June 2011.

[3] The CMS Collaboration et al. "The CMS experiment at the CERN LHC," J. Inst, vol. 3, S08004, pp. 1-334, 2008, doi: 10.1088/1748-0221/3/08/S08004.

[4] R. Pordes et al., "The open science grid," J. Phys.: Conf. Ser., vol. 78, 012057, pp. 1-15, 2007, doi: 10.1088/1742-6596/78/1/012057.

[5] "Open Science Grid Home page," http://www.opensciencegrid.org/, Accessed June 2011.

[6] "Glidein Frontend documentation," http://tinyurl.com/glideinWMS/doc.prd/frontend/configuration.html, Accessed June 2011.

[7] I. Sfiligoi et al., "Operating a production pilot factory serving several scientific domains," J. Phys.: Conf. Ser., in press.

[8] P. Andreetto et al., "The gLite workload management system," J. Phys.: Conf. Ser., vol. 119, 062007, pp. 1-10, 2008, doi: 10.1088/1742-6596/119/6/062007

[9] P. Mhashilkar, G. Garzoglio, T. Levshina, and S. Timm, "ReSS: Resource Selection Service for National and Campus Grid Infrastructure," J. Phys.: Conf. Ser., vol. 219, 062059, pp. 1-8, 2010, doi: 10.1088/1742-6596/219/6/062059

[10] "OSG MM - The Open Science Grid Match Maker," http://osgmm.sourceforge.net/, Accessed June 2011.

[11] T. Maeno, "PanDA: distributed production and distributed analysis system for ATLAS ," J. Phys.: Conf. Ser., vol. 119, 062036, pp. 1-4, 2008, doi: 10.1088/1742-6596/119/6/062036

[12] A. Tsaregorodtsev et al., "DIRAC: a community grid solution," J. Phys.: Conf. Ser., vol. 119, 062048, pp. 1-12, 2008, doi: 10.1088/1742-6596/119/6/062048

[13] E. Walker, J.P. Gardner, V. Litvin, and E.L. Turner, "Creating personal adaptive clusters for managing scientific jobs in a distributed computing environment," CLADE, 2006 IEEE, pp. 95-103, 2006, doi: 10.1109/CLADE.2006.1652061

 ISBN: 978-1-61208-156-4

A Case Study on Self-Sufficiency of Individual Robotic Modules in an Arena With Limited Energy Resources

Raja Humza
Department of Biomedical Microsystems
Fraunhofer Institute for Biomedical Engineering (IBMT)
St. Ingbert, Germany
humza.raja@ibmt.fraunhofer.de

Oliver Scholz
School of Engineering
University of Applied Sciences (HTW)
Saarbrücken, Germany
oliver.scholz@htw-saarland.de

***Abstract*—From self-sufficiency perspective, in an artificial robotic swarm, the critical parameter that influences the collective and individual's behavior is not the time required to locate and successfully dock to a recharge station in the arena, rather it is the time a robot occupies a recharge station to fully recharge its on-board batteries. It becomes critical because during recharging of a robot, the recharge station is no longer available for the rest of the modules in the arena. In a bigger swarm, it becomes impractical due to several reasons to deploy an equivalent number of recharge stations in the arena. Therefore, it is of great interest for the system designers to know the appropriate ratio between the number of recharge stations and the number of robotic modules. To test the behavior of an autonomous robotic swarm we have employed traditional bio-inspired techniques with a simple threshold based mechanism that uses the on-board state of charge of a robot to govern and adapt a robot's behavior in different scenarios. The paper concludes with the validation of initial work in Player/Stage simulator and a future work plan.**

***Keywords*-Autonomous robotic systems; trophallaxis; power management; self-sufficiency; charger to robot ratio; CRR.**

I. INTRODUCTION

Long term survivability and autonomy of an autonomous system (living and artificial) are governed by energy resources available in the environment and its ability to adapt itself to changing conditions. From the energy autonomy perspective, foraging as in nature, provides an ability to a mobile robotic system to be aware of its dynamically changing energy requirements in order to autonomously search and regain its replenished energy from the environment. Where as, the adaptiveness empowers a system to tune its behavior/operations with the internal and external system dynamics. The foraging principle has been applied in a variety of ways, to develop the control and behavior of a robotic swarm – both individually and collectively, e.g., collection of objects scattered around the arena and to assemble them in some random or a predefined location [1], [2], [3], and to investigate the collective behavior of a multi-robotic system [4], [5].

According to McFarland, to be energetically autonomous, the *self-sufficiency* is an ability of an autonomous system to maintain itself in a viable state for a longer period of time [6]. A self-sufficient robot therefore has the ability to perform the "basic cycle of work", i.e., find fuel and refuel itself [7].

In literature, to prolong the operational time and improve the energy autonomy of individual autonomous modules in a robotic swarm different techniques have been applied that use either, threshold mechanisms which are based on the battery voltage level [8], activation variables [9], [10], or time [11] to determine an appropriate action for a robot. Another approach that has been applied to prolong the activity time of autonomous robotic systems in a constraint environment includes "charge station sharing", as in [11], [12]. In the said approach, Michaud and Robichaud highlighted the potential issues that arise in an arena with limited energy resources, e.g., "*when is it appropriate for a robot to recharge*", "*how long should the robot recharge itself*", "*what can be done to preserve energy*". In their approach, the operational time estimated from the battery voltage is used to determine the appropriate "time" to recharge a robot.

This paper is organized as follows: Section II briefly presents the related work. Section III describes the addressed problem. Section IV then describes the proposed finite state machine that controls the behavior of a robot in different scenarios. Section V presents the strategies used to measure the overall system performance by varying swarm population and number of recharge stations. Section VI describes the experimental setup, and later presents and discusses the simulation results. At the end, Section VII draws a conclusion and presents future work plans.

II. RELATED WORK

In an approach of collective energy distribution to achieve long term survivability, Kubo and Melhuish [13] explored the idea of robot trophallaxis. Trophallaxis, which is a food sharing phenomenon found in nature, enables a robot to donate an amount of its internal energy reserve to its weaker (having less energy) fellow robots in the swarm. In their model, the robots that are engaged in a cleaning task share their energy between each other using a simple collision

 ISBN: 978-1-61208-156-4

based mechanism in which after a simple arbitration mechanism, one becomes energy "donor" and the other becomes the energy "recipient". The system performance was then measured using three energy transfer rules and by varying the number of robots with a static recharge station in the arena.

Recently, Kernbach et. al. in [14] presented a kinetic model of swarm foraging to maintain energy homeostasis in an arena with fixed recharge stations. As defined, energy homeostasis is a means of keeping the energy flow balance/equivalent among the individuals in a robotic swarm. Their model uses the time spent by the robots during *working*, *searching*, *waiting*, and *recharging*, to measure the energetic efficiency of the swarm. The model then assumes the charging and discharging time of the robotic modules to be approx. equal to the same charging and discharging currents of the *Jasmine* robot [15]. This means, while operating in the environment, one half of the swarm population keeps itself busy in performing the assigned task and the other half is docked to the recharge stations. The assumption in other words, sets the number of recharge stations to half the number of robots in the arena. To increase the energetic efficiency of the swarm in their work, they mainly focused on the time spent by the robots during "searching" and "waiting" for a recharge station rather than the "charging" and "discharging"(working) time periods.

III. Problem Statement

In an artificial robotic swarm, the parameters that control or effect the self-sufficiency of robotic modules include the energy availability in the environment, energy required to regain replenished energy, and the recharging time especially in case the energy resources are fewer than the number of robotic modules in the arena. From the system design perspective, the recharging time of an on-board battery pack, which is a critical parameter, usually depends on the battery chemistry. For example, the lead-acid battery requires roughly between 12–16 hours [16], whereas, the lithium-ion polymer cells require about 1–2 hours to fully recharge. Other parameters that depend on the battery chemistry and are relevant to the robot electronic and mechanical design include, charge/energy density, supply voltage and the size of the cells.

In an autonomous robotic swarm, the foremost objective is the long term survival of a maximum number of modules in a swarm, as in the case of the SYMBRION/REPLICATOR[1] project, which sets a grand challenge, namely 100 robots surviving 100 days. The fundamental questions that we have found unanswered in the literature includes: what type and what level of cooperation is beneficial for the group that maximizes the number of active modules in the arena? What is the effect of longer recharging time on the behavior and SOC of competing robots? In a real environment with a large number of robotic individuals, e.g. > 50 robotic modules, it is quite impractical to deploy a recharge station for every single robot in the arena, mainly, due to the cost factor involved in the production of the large number of recharge stations and the space requirement in the arena. Therefore, it is important to know how many recharge stations are minimally required to keep a sufficient number of robots alive for a longer period of time in a given environment. In other words, what should be a sensible charger to robot ratio (CRR), i.e.,

$$CRR = \frac{\text{Number of recharge stations}}{\text{Total number of robots}}. \qquad (1)$$

As already mentioned, Kernbach et. al. in [14] have proposed the ratio CRR to be 0.5 – the number of recharge stations equals half the number of robotic modules, to optimize the number of active modules in the arena. But with the assumption that the charging and discharging time of the robots are equal. Whereas, we want to find out the CRR value in a scenario where the recharging time is less than the average discharging/operational time of a robot – roughly half.

IV. Finite State Machine for Collective Foraging

To survive in an arena with fewer energy resources for a longer period of time, we have developed a finite state machine to control an individual's behavior in the swarm. Figure 1 shows the finite state machine (FSM) of an autonomous foraging robot. A robot in its life cycle goes through the following states: *"nesting", "trophallaxis", "searching", "approach and dock to a recharge station", "recharging", "avoidance", "waiting", "faint"*, and finally, *"dead"*. The transition between the states is triggered either on external stimuli (from sensor values), e.g. obstacles, or on internal stimuli, i.e. state of charge (SOC) of a robot. The state transitions that are triggered on internal stimuli use two variables: "current state of charge" (SOC_{curr}) and the "state of charge reserved" (SOC_{res}), which is considered to be the maximum amount of charge a robotic module can reserve in order to search and successfully dock to a recharge station in the arena. The current state of charge of a robot's battery pack in percentage is obtained as,

$$SOC_{curr} = \frac{\text{remaining charge}}{\text{maximum charge capacity}} \times 100. \qquad (2)$$

A. Nesting

It is the healthiest state of a robot in its life cycle. A robot in nesting state has enough energy to explore the environment, perform the assigned task and if required can donate its excess charge to any other distressed robot in the arena. A robot remains in nesting state until it has just

[1] www.replicators.eu

 ISBN: 978-1-61208-156-4

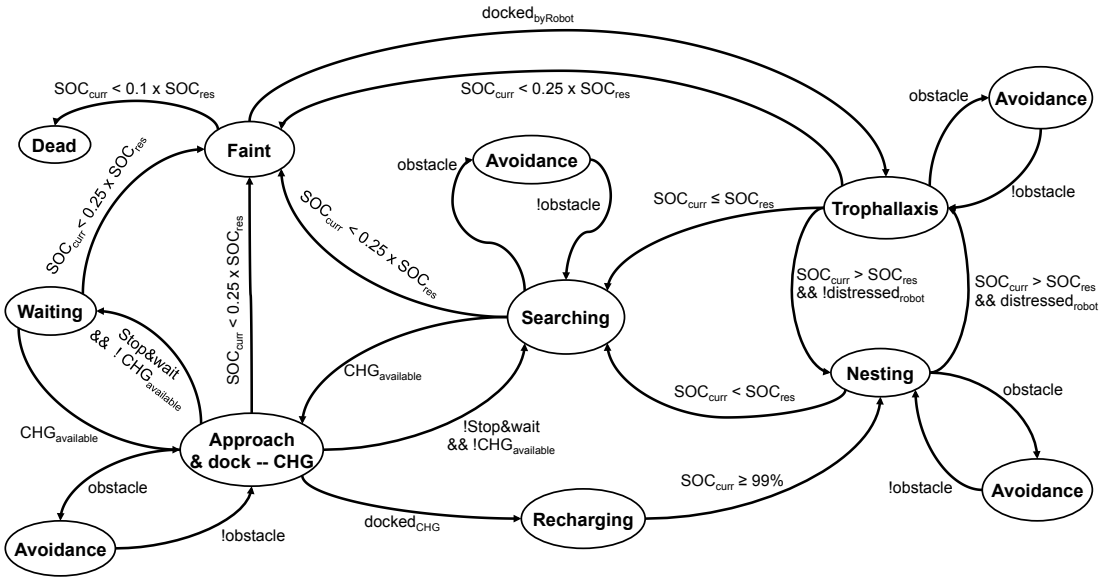


Figure 1: Finite state machine that controls the behavior of an autonomous robot

enough energy left to autonomously reach and dock to a recharge station without any external assistance, i.e.,

$$SOC_{curr} > SOC_{res}.$$

B. Searching

It is the state in which the highest priority task for a robot is to search/locate a recharge station in the arena to regain its replenished energy before it completely runs out of energy. A robot enters "searching" mode either from the "nesting" or "trophallaxis" state, when SOC_{curr} becomes less than or equals SOC_{res}. This means, when the condition

$$SOC_{curr} \leq SOC_{res}$$

becomes true, a robot spends its remaining energy for its survival. For further clarity, the variable SOC_{res} is just a threshold that assigns the foraging task as the highest priority task for a robot, when the above condition becomes true.

C. Avoidance

An obstacle can either be a robot, a recharge station or a boundary wall in the arena, within the detection range of a robot, which forces it to enter the "avoidance" state for the period of time the obstacle remains in its detection range.

D. Trophallaxis

This is a bio-inspired phenomenon employed here to increase the survival time of individual modules in a robotic swarm. Upon receiving distress messages from a faint robot a healthy, "nesting" robot in the vicinity automatically aligns and docks itself to it for the purpose of energy donation. After successfully docking, the "healthy/donor" robot donates its excess energy, i.e., $SOC_{curr} - SOC_{res}$, to the faint robot. After energy exchange, the robot in "faint" state remains in this state if its SOC_{curr} remains insufficient to start searching the recharge station in the arena again, i.e.,

$$SOC_{curr} < 0.25 \times SOC_{res}.$$

E. Approach and Dock

Upon detecting either a recharge station or a distressed robot in the arena, a robot "searching" or "nesting" automatically aligns itself towards its target for the purpose of docking. In case multiple robots approach the same target, a recharge station or a distressed robot, those that succumb return back to their previous state. On successfully docking with the recharge station, a robot enters "recharging" state.

F. Recharging

As mentioned earlier, in an arena with limited energy resources the recharging time of a robot becomes critical as the recharge station becomes unavailable to the rest of the competing robots in the arena. In case of a REPLICATOR robotic module, that uses a 6 cells lithium polymer battery pack, the recharging time is roughly between 1–1.5 hours at a continuous current corresponding to 1C, i.e., 1400 mA.

Likewise, in simulation, ignoring the time required for cell balancing, a robot in the state of "recharging" occupies the recharge station no longer than for a duration of approximately 1 hour, depending on its battery SOC_{curr}.

G. Waiting

A robot enters the "waiting" state, if it stops receiving beacon signals from the recharge station while in the "approach and dock" state. This happens in two scenarios: either when a competing robot in the arena successfully docks to

 ISBN: 978-1-61208-156-4

the recharge station prior to it, or the robot itself moves away from the recharge station, so that it misses the line of sight. A robot remains in waiting state until it starts receiving charger signals again, or on the expiry of a waiting timer – T_w. The length of T_w is in fact the amount of time required to fully recharge a depleted battery pack at 1C. In our implementation, we set it to 10 minutes to compensate the frequent/erroneous triggering into "waiting" state, e.g., moving away form the line of sight.

H. Fainting

It is the state in which the robot's remaining SOC is considered as not enough to search and successfully dock to a recharge station in the arena. A robot in this state therefore stops moving and starts broadcasting "distress messages" on its IR communication channels. In the current framework, a robot remains in faint state until

$$0.1 \times SOC_{res} < SOC_{curr} < 0.25 \times SOC_{res}.$$

I. Dead

It is considered to be the end of the life cycle of an autonomous robot. It is reached from the "faint" state in case the distressed robot does not receive any aid from the fellow robots in the arena. It is reached when the condition

$$SOC_{curr} \leq 0.1 \times SOC_{res}$$

becomes true.

V. Foraging Strategies

To estimate a tradeoff between the number of recharge stations and robotic modules, we have devised some simple strategies to measure and compare the performance of a robotic swarm in different scenarios.

A. Simple

This scenario was in fact devised to compare and evaluate the results obtained from other techniques. In the simple scenario, the individual autonomous robots lack the ability of dynamic collaboration, i.e., trophallaxis. In fact, they are selfish in the respect that each module is concerned with its own survival in the arena. In this approach, 70% of the max. SOC is considered as SOC_{res}.

B. Trophallaxis

Trophallaxis provides an individual the ability to sense and dynamically cooperate with fellow robots in the arena. With this ability, a healthy, "nesting" robot becomes a donor by sharing its excess energy as soon as it docks to a distressed robot that resides in the "faint" state. Upon a successful docking it starts donating its excess charge to it, where the excess amount of charge "E" is calculated as the difference between SOC_{curr} and SOC_{res}, i.e.

$$E = SOC_{curr} - SOC_{res}. \quad (3)$$

C. Learn and Adapt with Trophallaxis

In the above two foraging strategies, i.e., "simple" and "trophallaxis", the variable SOC_{res} uses a fixed value, i.e. 70% of max. SOC, for the transition between the FSM states. In the case described here, with the ability to learn and adapt, each robot in the swarm learns from the environment an appropriate value of SOC_{res} each time it successfully docks to a recharge station from the time it started its search, to update its previous estimate. Let $SOC_{res}(n-1)$ be the value learned at docking instance $(n-1)$ with a recharge station, and $SOC_{res}(n)$ be the reserved SOC learned at current docking instance (n), i.e. the new value. Where, $n = 0, 1, 2, ...$ The SOC_{res} for the next run was then updated by taking the average of the two estimates, as,

$$SOC_{res}(n) = \frac{SOC_{res}(n-1) + SOC_{res}(n)}{2}, \quad (4)$$

and, by scaling it,

$$SOC_{res}(n) = SOC_{res}(n) \times 1.5, \quad (5)$$

with a factor of "1.5". The scaling was done to reduce the effect of the low value of SOC_{res} learned in the two estimates. Otherwise, an abrupt decrease in the SOC_{res} that reflects the energy abundance in the environment, may lead to a false prediction.

D. Stop and Wait

It's a simple technique that is based on the idea of energy conservation. With the "stop and wait" ability, if a robot stops receiving "charger" signals when in the "approach and dock" state, instead of switching back to "searching" state again, it enters the "waiting" state with the assumption that the charger is occupied by some other robot. A robot remains in "waiting" state until it starts receiving "charger" signals again or the waiting timer (T_w) expires. During waiting state, a robot stops its locomotion to save its energy while the rest of the on-board electronics remain active. Further, in the "stop and wait" topology, a robot disables its trophallaxis ability and strives only for its own survival.

E. Stop and Wait with Trophallaxis:

It is meant to see the combined effect of two simple techniques on energy distribution in the swarm. In this topology, the robots in the swarm on one side preserve their energy by actively choosing the "wait" state and on the other side donate their excess energy to fellow robots to increase the number of active robots in the arena.

VI. Experimental Setup and results

At this stage of development, to explore the dynamic behavior of robotic modules with limited sensing abilities in the arena, we implemented a simple robot model and the test scenarios in Stage simulator [17]. For this purpose, we were provided with an abstract model of a REPLICATOR robot

 ISBN: 978-1-61208-156-4

in the "Stage" simulator by Wenguo Liu from the Bristol robotic laboratory (BRL), University of the West of England, Bristol, UK.

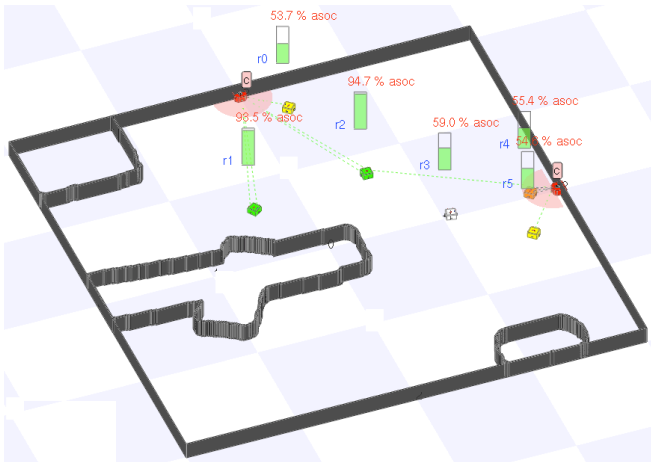


Figure 2: A screen shot of an example scenario in Stage simulator: 6 robots with 2 recharge stations in a simulation arena of dimensions 4m × 4m

A. Robot Model

The dimensions of the simulated robot model are 80 mm × 80 mm × 80 mm. It has four infra-red (IR) sensors (one on each side) for communication in the arena, four status LEDs (one on each side), eight light sensors (two on each side), four docking units (one on each side), a locomotion drive, a battery pack, and a robot controller. The IR and light sensors help the robot to perform beacon detection in order to align itself towards the target. The on-board partially simulated battery pack as in the original robot, provides the nominal voltage of 22.2 V with maximum 1400 mAh of charge capacity. Considering 75 % efficiency of the LiPo battery pack, it finally provides a max. capacity of 1050 mAh. The battery management module uses a coulombs counting approach for the estimation of battery capacity and the absolute state of charge. The simulated robot current consumption values are also taken from empirical values recorded with real robot hardware. On every time instant "*t*", the on-board electronics current consumption was taken randomly between 25 mA to 50 mA at nominal voltage. During random walk the robot's locomotion drive consumed a continuous current of 250 mA, whereas the docking unit that becomes active only during docking and undocking with a recharge station or to a robot for a period of 5 seconds as in the original system, drew a current of 55 mA per second.

B. Simulation Results

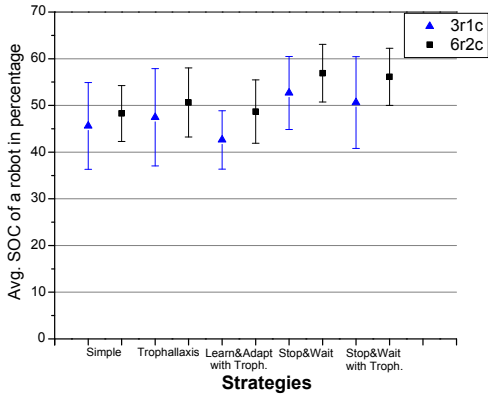
To explore the foraging behavior of a robotic swarm, collectively and individually, with limited energy resources we have varied the number of robots with fixed number of recharge stations in a simulation arena of 4m × 4m. Figure 2 shows a screen shot of an example scenario in Stage simulator [17] with 6 robotic modules and 2 fixed recharge stations.

In our experiments we are focusing on the average state of charge of robotic modules as it provides a rough estimate of flow of energy and number of dead robots in the arena. The higher the energy level of the robotic individuals the longer they can survive on their own which in turn enables them to perform a variety of tasks collectively and individually in the arena. Therefore, to get a reliable estimate of average SOC of a robot and the average number of dead robots in a given condition, we ran each strategy, as mentioned in section V, 10 times with each robot having an initial SOC of 80 % of the max. battery capacity. Each time the simulation was then recorded for a period of 10 hours.

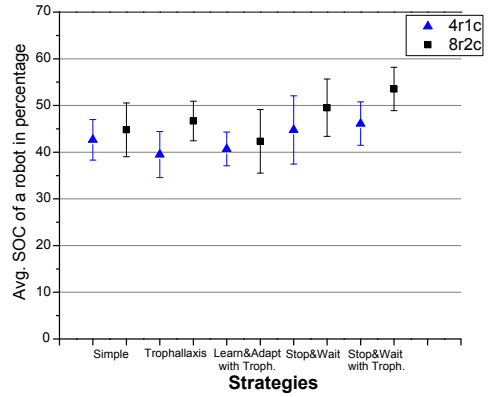
Figure 3 shows the simulation results in three cases recorded by varying the number of robots in the arena with a fixed number of recharge stations. In figure 3, the horizontal axis shows the applied "strategies", and the vertical axis shows the "average SOC of a robot" in percentage – averaged over 10 simulation runs. The error bars show the standard deviation of 10 simulation runs. Figure (3a) shows the average SOC of a robot, to compare the effect of limited energy resources in two scenarios: 3 robots with 1 recharge station (3r1c) and 6 robots with 2 recharge stations (6r2c), thus keeping the ratio CRR constant, i.e. 0.333. Figure (3b) shows the average SOC of a robot with CRR = 0.25 in two scenarios: 4 robots with 1 recharge station (4r1c) and 8 robots with 2 recharge stations (8r2c). And lastly, fig. (3c) shows the average SOC of a robot with CRR = 0.166 again in two scenarios: 6 robots with 1 recharge station (6r1c) and 12 robots with 2 recharge stations (12r2c). Before discussing the results shown in figure 3, to get an idea of the applied strategy on the swarm it is important to see the average number of dead robots at the end of simulation runs. Figure 4 shows the average number of dead robots in different scenarios with CRR = 0.333, 0.25, and 0.166. The horizontal axis shows the applied "strategies", and the vertical axis shows the "average number of dead robots". The error bars again show the standard deviation of 10 simulation runs.

In the case CRR = 0.33, the trophallaxis feature in a smaller swarm size neither contributed in increasing the avg. SOC of robots nor in decreasing the average number of dead robots. But on the contrary, in the bigger swarm it improves the avg. SOC of robots and slightly decreases the avg. dead robots. The "learn and adapt" strategy is barely able to produce the same results as with the simple strategy. The "stop and wait" has shown its effect only in the bigger swarm. In the case CRR = 0.25, the trophallaxis feature likewise in the smaller swarm fails to show it presence on the avg. SOC of robots. But the stop and wait strategy together with trophallaxis feature in the bigger swarm decreases the avg. no. of dead robots by increasing the avg. SOC of the

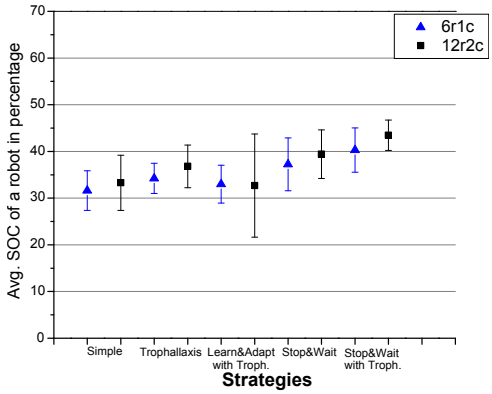
 ISBN: 978-1-61208-156-4



(a) CRR = 0.33

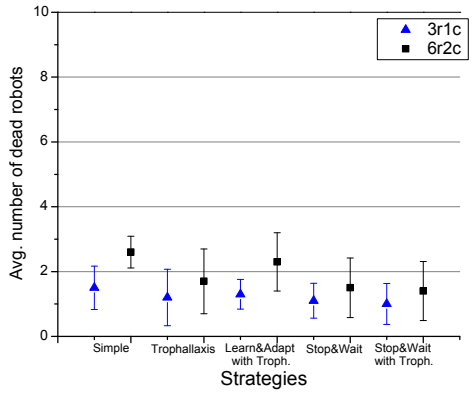


(b) CRR = 0.25

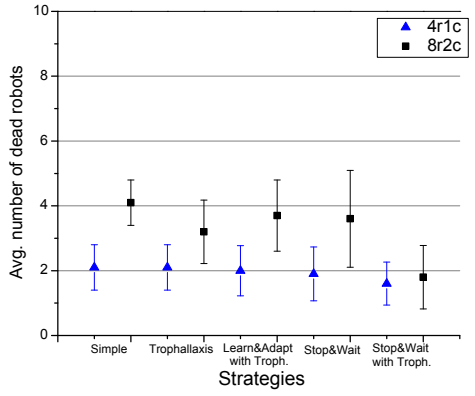


(c) CRR = 0.166

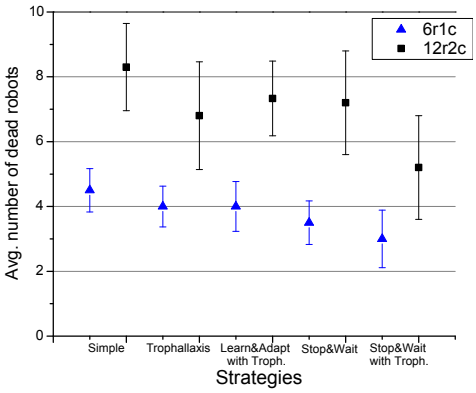
Figure 3: Average state of charge (SOC) of a robot in different scenarios with CRR = 0.33, 0.25, and 0.166. The error bars show the standard deviation of 10 simulation runs.



(a) CRR = 0.33



(b) CRR = 0.25



(c) CRR = 0.166

Figure 4: Average number of dead robots in different scenarios with CRR = 0.33, 0.25, and 0.166. The error bars show the standard deviation of 10 simulation runs.

robots in the swarm. At the end, in the case CRR = 0.166, we have seen nearly the similar trend in avg. SOC of robotic modules and the avg. number of dead robots as we have in case of CRR = 0.25.

Concerning the active number of robots, it is evident from figure 4, that the "trophallaxis" feature alone especially in the bigger swarm decreases the avg. number of dead robots in the arena and its effect is also comparable with the "stop and wait" strategy. Therefore, the combination of the two strategies showed a significant decrease in the avg. number of dead robots, especially in case of CRR = 0.25 and 0.16.

VII. CONCLUSION AND FUTURE WORKS

A. Conclusions

In this paper, we have explored the effect of few potential issues in a simulated environment that may become critical

for a robot swarm operating in a real environment with limited energy resources. For this purpose in our preliminary work, we have implemented a basic robot model and the test scenarios in Player/Stage simulator. The performance of the system is then evaluated from the avg. SOC of robotic modules and the avg. number of dead robots in the arena. Considering the two measures, we have seen that the CRR = 0.25 provides us a fair trade off between the number of recharge stations and the number of robotic modules in a closed environment. As in a bigger swarm, the robotic individuals are able to achieve nearly the same avg. SOC as in case of CRR = 0.33. This in fact provides us the motivation to test the system behavior with a bigger swarm size with the ratio CRR = 0.25.

B. Future Works

In the current framework, a critical parameter that is left and requires the attention of the system designer is the system energy loss especially during energy trophallaxis. In future, along with the simulation work we plan to implement and test such simple strategies on a real robot swarm, i.e., REPLICATOR robotic modules, to compare it with simulation results. The testing on the real hardware also involves the measurement of energy losses during trophallaxis and power sharing between the robots.

ACKNOWLEDGMENT

We are very thankful to “Wenguo Liu” from Bristol robotic laboratory (BRL), University of the West of England, Bristol, UK, and “Lachlan Murray” from University of York for providing their assistance in Stage simulator. This work is part of a European Union funded project named “REPLICATOR”. The “REPLICATOR” project is funded within the work program Cognitive Systems, Interaction, Robotics under the grant agreement no. 216240.

REFERENCES

[1] M. J. Matarić, “Minimizing complexity in controlling a mobile robot population,” in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, (Nice, France), pp. 830–835, 1992.

[2] E. Nitz, R. C. Arking, and T. Balch, “Communication of behavioral state in multi-agent retrieval tasks,” in *Proceedings of the 1993 IEEE International Conference on Robotics and Automation* (L. W. Robert and O. Conner, eds.), vol. 3, (Atlanta, GE), pp. 588–594, IEEE Computer Society Press, May 1993.

[3] D. Goldberg and M. J. Matarić, “Robust behavior-based control for distributed multi-robot collection tasks,” *CiteSeerX - Scientific Literature Digital Library (United States)*, 2000.

[4] L. Parker, “The effect of action recognition and robot awareness in cooperative robotic teams,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, p. 212, 1995.

[5] K. Sugawara and T. Watanabe, “A study on foraging behavior of simple multi-robot system,” in *IECON 02: IEEE 2002 28th Annual Conference of the Industrial Electronics Society*, vol. 4, pp. 3085 – 3090, Nov. 2002.

[6] D. McFarland, “Autonomy and self-sufficiency in robots,” in *The Artificial Life Route To Artificial Intelligence: Building Embodied Situated Agents* (L. Steels and R. Brooks, eds.), (Lawrence Erlbaum, USA), pp. 187–213, 1994.

[7] D. McFarland and E. Spier, “Basic cycles, utility and opportunism in self-sufficient robots,” *Robotics and Autonomous Systems*, vol. 20, no. 2-4, pp. 179 – 190, 1997. Practice and Future of Autonomous Agents.

[8] S. Oh and A. Zelinsky, “Autonomous battery recharging for indoor mobile robots,” in *Australian Conference on Robotics and Automation*, 2000.

[9] F. Michaud and j. Audet, “Using motives and artificial emotions for long-term activity of an autonomous robot,” in *Proceedings of the fifth international conference on Autonomous agents*, AGENTS ’01, (New York, NY, USA), pp. 188–189, ACM, 2001.

[10] F. Michaud, E. Robichaud, and J. Audet, “Using motives and artificial emotions for prolonged activity of a group of autonomous robots,” in *Emotional and Intelligent II: The Tangled Knot of Social Cognition - AAAI Fall Symposium, Cape Code Massachussetts*, pp. 85–90, 2001.

[11] F. Michaud and E. Robichaud, “Sharing charging stations for long-term activity of autonomous robots,” in *Proc. of the International Conference on Interlligent Robots and Systems, IEEE/RSJ*, (EPFL, Lausanne, Switzerland), pp. 2746–2751, Oct. 2002.

[12] F. Sempe, A. Munoz, and A. Drogoul, “Autonomous robots sharing a charging station with no communication: a case study,” in *Distributed Autonomous Robotic Systems* (H. Asama, ed.), vol. 5, (Tokyo), pp. 91–100, Springer-Verlag, 2002.

[13] C. Melhuish and M. Kubo, “Collective energy distribution: Maintaining the energy balance in distributed autonomous robots using trophallaxis,” in *Distributed Autonomous Robotic Systems 6* (R. Alami, R. Chatila, and H. Asama, eds.), pp. 275–284, Springer Japan, 2007.

[14] S. Kernbach, *Handbook of Collective Robotics*, ch. Improving the Scalability of Collective Systems, pp. 225–256. Springer-Verlag Berlin, 2010.

[15] S. Kornienko, O. Kornienko, C. Constantinescu, M. Pradier, and P. Levi, “Cognitive micro-agents: individual and collective perception in microcorob swarm,” in *IJCAI-05 Workshop on Agents in Real-Time and Dynamic Environment*, pp. 33–42, 2005.

[16] “Charging the lead acid battery.” http://batteryuniversity.com/learn/article/charging_the_lead_acid_battery/, [Last accessed: 1 Apr. 2011].

[17] B. Gerkey, R. Vaughan, and A. Howard, “The player/stage project: Tools for multi-robot and distributed sensor systems,” in *11th International Conference on Advanced Robotics (ICAR 2003)*, (Coimbra, Portugal), pp. 317–323, June 2003.

 ISBN: 978-1-61208-156-4

Dependable and Usage-Aware Service Binding

Holger Klus, Dirk Niebuhr, Andreas Rausch
Department of Informatics
Clausthal University of Technology
Clausthal-Zellerfeld, Germany
Email: {holger.klus, dirk.niebuhr, andreas.rausch}@tu-clausthal.de

***Abstract*—The internet is evolving from a global information network to an environment that offers services for all areas of life and business, such as virtual insurance, online banking, or entertainment. Such services are created frequently during runtime by service providers according to specific user needs and they operate in a network and service environment that provides unified access to virtualized resources. Since the number of available services increases rapidly, it is hard for a client to find appropriate services and to compose them to useful systems. Additionally, clients may use the services in changing situations and the set of appropriate services and their binding should change accordingly to meet the users' needs in every situation.**

In this article, we present an approach which enables the automatic context-aware binding of services during runtime to so called dynamic adaptive systems. For this purpose, we introduce an approach for checking semantical compatibility [1] of services followed by an integrated approach for usage-aware service binding. Finally, we present our infrastructure DAiSI which provides an integrated implementation of both aspects. DAiSI has been applied to an application prototype which is also presented in this article and which validates the applicability of the presented solutions.

***Keywords*-Service orchestration, service binding, dynamic adaptive systems, runtime testing, context awareness, runtime adaptation, user interaction**

I. Introduction

Software-based systems pervade our daily life – at work as well as at home. Public administration or enterprise organization can hardly be managed without software-based systems. We come across devices executing software in nearly every household. Increasing size, penetration and features of software-based systems have brought us to a point, where software-based systems are the most complex systems engineered by mankind.

Current research areas, like ubiquitous computing, pervasive computing, or ultra-large scale systems, want to enable the engineering of future software-based systems by sharing a common trend: Complex software-based systems are no longer considered to have well-defined boundaries. Instead future software-based systems are composed of a large number of distributed, decentralized, autonomous, interacting, cooperating, organically grown, heterogeneous, and continually evolving services. Adaptation, self-x-properties, and autonomous computing are envisaged in order to respond to short-term changes of these service-based system itself, the context, or a user's expectation. Furthermore, to cover the long-term evolution of service-based systems becoming larger, more heterogeneous, and long-lived, service-based systems so called dynamic adaptive service-based systems must have the ability to continually evolve and grow, even in situations unknown during development time.

Since the number of available services increases rapidly, the requirements of a service user – independent of the service user seen as human end-user or as another service requesting for services – regarding other services get even more and even harder to manage. As a result, a service user has to solve similar problems regarding service discovery, like the cumbersome and painful task to find information in the Internet with different web search engines and other, more or less sophisticated tools currently available. It is still the user (human or machine) who has to be the active part searching and browsing the World Wide Web like looking for a needle in a haystack. Similar problems arise in the Internet of Services: The service user has to be the active part searching in the Internet of Services for the best services. Once he has found service candidates, it's again up to him to bind and orchestrate these services to the intended dynamic adaptive service-based system. The following questions arise regarding this process:

- What services provide (in a dependable manner) the service user's required service functionality and quality?
- What services fit best to the actual service user's context?
- What services fit best to the actual service user's usage intention?
- What is required by the selected services?
- How can the service binding be established for a dependable and usage-aware orchestration?

The aforementioned characteristics of a dynamic adaptive service-based system are the reason that classical software engineering approaches need to be enhanced. In traditional software engineering many aspects like software architecture, testing, component configuration were considered during design time. In dynamic adaptive service-based systems several aspects, like for instance the service selection, binding and orchestration to the intended dynamic adaptive service-based system, can only be processed during runtime considering context information and service users' intentions.

In the following we will introduce our means to enable a (semi-)automatic binding of dynamic adaptive service-based systems. Thus we will first introduce the characteristics of dynamic adaptive service-based systems regarding service bindings. We will continue by giving an example of a dynamic adaptive service-based system in section III which is used to illustrate our methods of establishing a service binding in dynamic adaptive service-based systems. Section IV introduces our approach to achieve dependable, usage-aware service bindings by applying runtime-testing and considering the context during service binding. A conclusion rounds up the paper.

[1] Patent pending. Patent Nr. 10 2008 050 843.8, 8.10.2008

 ISBN: 978-1-61208-156-4

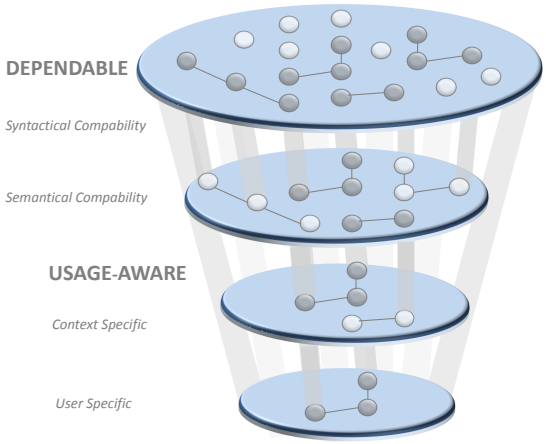


Figure 1. Dependable And Usage-Aware Service Binding

II. Dynamic Adaptive Service-Based Systems from a Bird's Eye View

The Internet of Services offers a frequently dynamically changing set of services for all areas of life and business. To fulfill the actual user's[2] needs with respect to it's context the proper dynamic adaptive service-based system has to be created from services contained in the cloud of services. Technical infrastructures for such a cloud of services are already available, like for instance OSGi [1] or SOPERA [2].

Using these infrastructures one has to manually define, select, bind and orchestrate required services respectively one has to implement a predefined set of possible configurations. Assume – due to continual evolution and growth of number of existing services in the cloud – a new, yet not considered, service appears, which would perfectly fit as an additional configuration option into the defined set of possible service binding and orchestration configurations. As long as the manually defined respectively implemented set of possible configurations will not be re-defined respectively re-implemented this new service will not be taken into consideration during selection, binding and orchestration of the services in the cloud to the desired dynamic adaptive service-based system.

For that reason, more sophisticated infrastructures have been developed recently in research and practice, like for instance Microsoft Extensibility Framework (MEF) [3], ASG [4], KER-META [5], or DAiSI [6]. These infrastructures provide support for the top level shown in Figure 1: Infrastructures like for instance MEF or DAiSI introspect the provided and required interfaces of existing services and service users in the cloud and try to automatically bind those that match on the syntactical level.

Obviously, to provide dependable dynamic adaptive service-based systems, guaranteeing syntactical compatibility in service binding is not enough. Therefore one also has to guarantee the semantical compatibility of a service binding more precisely a compatible behavior of a service user and its required service to be bound. Hence as shown in Figure 1 in the next level – semantical compatibility – only those required and provided services will be bound that have a compatible syntax and behavior.

As a result we can guarantee the dependability of a service binding although it will be established automatically during runtime. However we have to go further. It is not enough to establish only correct bindings: we also have to take care that only those service bindings that match with the actual context and an actual user's intention will be established. For that reasons on the next level, the context specific level, only bindings between those services are allowed that match with the actual context. And finally, to provide dependable and usage-aware service bindings, the last level selects those bindings from the remaining possible bindings, which match the current user's intention.

By applying this four level approach to reduce possible service bindings finally only dependable and usage-aware bindings are established. In the next section we will show these four levels within a concrete application scenario. We use it to illustrate the infrastructure-related challenges that we face, to enable dynamic adaptive service-based system with all characteristics described above.

[2]Note, users can be human users as well as other services requesting services from this service cloud.

Figure 2. Sequence of Events During an Disaster

III. Scenario

Imagine a huge disaster like the one, which occurred during an airshow in Ramstein in 1988. Two planes collided in air and crashed down into the audience. In cases of such a disaster the number of casualties exceeds the number of medics by far. Thus medics need to get a quick overview of the whole situation before treating individuals. Therefore they do a quick triage [7], classifying the casualties regarding the severity of their injury, in order to treat casualties with serious injuries first. The sequence of events in case of such a disaster is depicted in Figure 2.

In our scenario, medics are supported by an IT system, enabling them to get a quick overview of the overall situation and to keep track of the physical situation of previously classified casualties. Medics come along with a medic unit and a bunch of casualty units. While a medic unit is a smartphone in our scenario, a casualty unit consists of vital data sensors. Each medic unit runs a medic application and each casualty unit runs a casualty service.

Whenever medics discover a casualty in field, they equip him with a casualty unit and thus the according casualty service of this casualty unit is started. The medic enters data regarding the casualty like his identifier, name, gender, or current position on a medic application. The data entered by the medic is stored by the casualty service.

 ISBN: 978-1-61208-156-4

Biosensors like pulse rate sensors or blood pressure sensors are part of a casualty unit in order to keep the triage class of the associated casualty up to date if his condition changes over time. Thus a casualty service executed on a casualty unit enables medics to capture and monitor a casualty's physical condition without needing to be physically present at his place. Within our example implementation, we used small sensor nodes [8] to realize casualty units.

After finishing the triage process, medics can use their medic applications to locate nearby casualties (respectively their services) or those, which need help most urgently. Information about these casualties is displayed by the medic application. This information is retrieved from the casualty services.

Figure 3 shows an excerpt of the domain architecture used for the emergency management system in our scenario. You can see, that casualty services provide a service interface CasualtySIf, which is used by medic applications. This interface provides access to the vital data sensors and enables service users to read and update the triage class of the associated casualty.

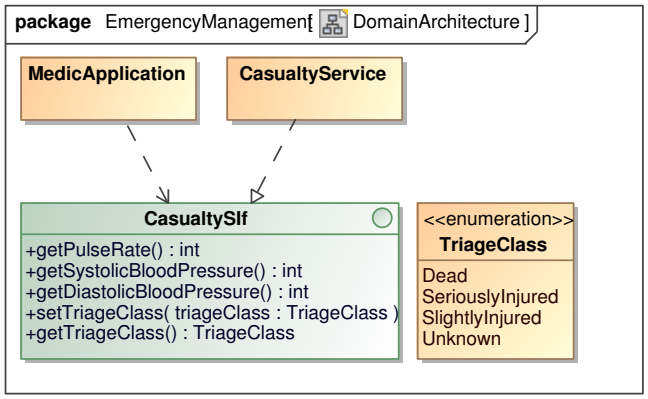


Figure 3. A Domain Architecture for Emergency Management Systems

We distinguish between two kinds of medics. One group of medics is responsible for the evacuation of slightly injured casualties while the other group takes care of seriously injured casualties. Casualties belonging to the aforementioned groups are displayed on the smartphones of the according medics.

Given that only casualties in the direct vicinity of a medic should be taken into consideration, only those are displayed on a medics' smartphone. Consequently medic get a filtered view of the situation.

This system is a typical dynamic adaptive service-based system. It consists of a vast array of services which are bound to service users at runtime: casualty services and medic applications. All services respectively service users can be provided by different vendors. Thus not each medic applications is compatible with each casualty service; To avoid malfunctions only compatible medic applications and casualty services may be bound to each other.

The overall system is evolving during runtime as new casualties may be integrated via their casualty service as well as casualty services may leave the system as casualties are transported to hospitals for further treatment. Next to this, medics acting as service users may enter or leave the system at any time.

We use a map depicted in Figure 4 to give you an overview of the situation which will be considered in the following. The map depicts all casualties, medics and their locations.

The map is divided into three areas: Area 1 is assigned to medic M1, area 2 is assigned to M2 and consequently area 3 is assigned to medic M3. The casualties in our example are classified into two triage classes: slightly injured casualties and seriously injured casualties. Slightly injured casualties are colored in blue, like casualty e, whereas seriously injured casualties are colored in red, like a, b, and c. According to the triage class, medics are responsible for a subset of these casualties. To keep it simple in our example all medics are responsible for seriously injured casualties.

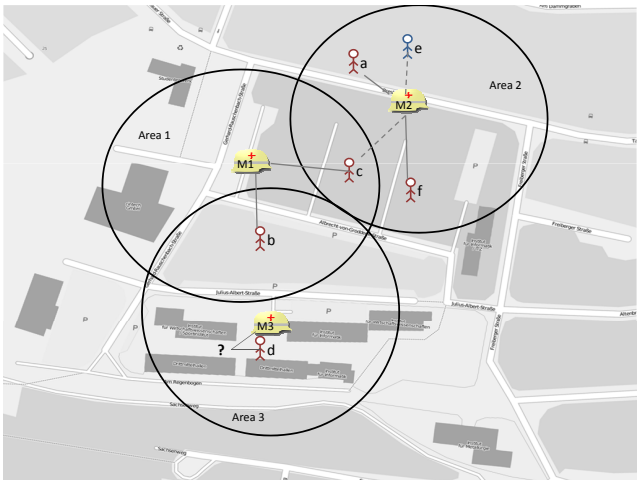


Figure 4. A Situation from our Application Scenario

Area 1 is allocated to M1; consequently he is responsible for casualty b and casualty c.

A closer look at area 2 points out, that medic M2 has the option to treat either casualty a or casualty f. Casualties c and e are not bound to his medic application. Casualty c is not bound to his medic application as he is semantically incompatible to M2. Casualty e is slightly injured and therefore not bound to his medic application as M2 is responsible for seriously injured casualties only.

Area 3 shows the scope of responsibility of M3. A characteristic of M3 needs to be considered: M3 is a medic, who has been at the disaster location by coincidence. He got slightly injured and now has to decide if he either wants to be treated as a casualty d or if he wants to help casualties as medic M3. The latter results in the fact that b is displayed on M3's medic application.

Summing up, casualty services can be bound to medic applications in different ways. This article describes the mechanisms to derive meaningful bindings. These mechanisms can be used to bind dependable and context aware systems at runtime.

IV. Challenges and Approaches in Dynamic Adaptive Service-Based Systems

In the following we describe the challenges associated with dynamic adaptive service-based systems. We illustrate these challenges by the means of our scenario introduced before and depict possible approaches.

Thereby we focus on challenges associated with providing an infrastructure for dynamic adaptive service-based systems which is capable of automatic system configuration.

 ISBN: 978-1-61208-156-4

A. Semantical Compatibility

During the scenario description we already identified several dimensions that may be considered to restrict the set of possible system configurations towards a reasonable size. The first dimension is the syntactic and semantic check.

1) Challenge: Dynamic adaptive service-based systems have to handle the fact that service providers as well as service users from various vendors may enter and leave the system at runtime. To allow service users to be bound to service providers at runtime, common domain architectures containing service interface specifications are provided. This causes the problem that even if all service providers adhere to this domain architecture, the system correctness cannot be guaranteed if service bindings are established which have not been tested in advance. This has two main causes:

1) A service provider implements a service interface in an incorrect way (*Incorrect service*).
2) A service user has implicit assumptions about an service exceeding the specification from the domain architecture (*Underspecification*).

Both causes are relevant in practice. On the one hand we may face incorrect services at any time, as we have an open system. On the other hand we need semantic underspecification in a domain architecture to enable vendors to provide service providers and service users with unique features.

Each service user may be tested respectively verified in combination with a specific service provider or even with various service providers. Once a service user is bound to a new service provider his assumptions might no longer hold resulting in an incorrect system configuration. Hence, the dependability of the resulting system can no longer be guaranteed, resulting in potential system failures. It is thus necessary to make additional requirements of service users explicit in a way enabling us to evaluate, whether a specific service provider fulfills them at runtime.

2) Approaches: The basic assumption of our work is, that the correctness of all possible system configurations of dynamic adaptive service-based systems cannot be verified respectively tested in advance. However, correctness of a specific binding between a service user and a service provider with respect to the service interface can be verified in advance. Hence, we still have to detect and prevent incorrect service bindings at runtime if we want to establish bindings which have not been tested in advance.

Our approach is the first step towards *dependable* dynamic adaptive service-based systems – since it is a testing approach we cannot *guarantee* the correctness of the system. Therefore it needs to be supplemented with a verification mechanism capable of proving a restricted set of safety-critical properties of the system at runtime.

The main question addressed by our approach is, whether a service provider is compatible to a service user. There are several approaches to provide (limited) statements regarding the correctness of service bindings:

- Enriching the domain architecture and its service interface specifications towards a complete specification. Therefore not only the service interfaces but also the required environment has to be specified in a semantically sound manner. This would mean, that the domain architecture contains a specification, which implies a single specific realization and leaves no space for unique features provided by single vendors. Consequently, there is no need for additional verification of the correct service binding during runtime as no different variants are possible. Hence, this is not a practical approach.
- If service user respectively service provider specify – not on the domain architecture but on the vendor specific implementation level – their required respectively provided service, they are compatible in general, if the specification of the provided service implies the specification of the required service [9]. As a consequence we have to prove the refinement relation between two specifications to ensure their compatibility. It is well known that for a specification technique with a high expressiveness (e.g. first order logic, turing machines) this can not be automatized [10], [11].
  If we limit the expressiveness of the used formalism (e.g. finite state machines, regular expressions, some forms of temporal logic) this relationship can automatically be proven [12], [13]. Following this approach, correctness cannot be guaranteed as we cannot specify some aspects – usually the critical ones – and therefore cannot verify them anymore.
- Using pre- and postconditions of the domain architecture to generate assertions, which are evaluated at runtime. This enables us to shut down the system, whenever an incompatibility of service bindings occurs. However we would like to be able to recognize these incompatibilities in advance and find an alternate service binding, enabling a seamlessly continued execution of the system.
- Checking the correctness of service bindings by bisimulation [14]. In this case we need to compare the states of two simulated system executions *for every system execution step*: one system is containing the service user and performs changes to its system state as specified in the required service specification, the other one is containing the service provider and performs changes to its system state as specified in the provided service specification. Beside the obvious massive performance problems in case of a recognized incompatibility the only solution is again shutting down the whole system.

Since none of the previously mentioned approaches is applicable at runtime very well, we propose integrating a runtime testing approach into a service orchestration infrastructure. Whenever a service user and a service provider should be bound together, test cases are executed in advance, which check, whether they match. Due to changing states of a service user / provider, additional tests need to be executed at system runtime as well. We need to show, that these test cases executed during reconfiguration are good enough to expose mismatches of service bindings. Whenever a test case fails, the blamable service binding can be deactivated and the remaining system can still be executed.

In the following we will consider service bindings between medic applications and casualty services from our scenario introduced before. In this scenario the casualty services act as service providers implementing the service interface CasualtySIf.

As already discussed in the previous sections, dynamic adaptive service-based systems, like the emergency assistance system in our application example, are based on a standardized domain architecture containing interface specifications as shown in Figure 3, For the application example this domain architecture needs to contain **interface** CasualtySIf. Medics use this interface to provide their functionality.

Since we want to build a dependable system binding service users and service providers developed by different vendors, the domain model may not only contain syntactical information like method signatures or datatypes occurring in the interfaces. It also may contain semantic specifications following the ‘Design by

Contract' [15] approach. To specify pre- and postconditions and invariants we may use mature specification techniques like the Java Modeling Language (JML) [16].

The specifications for **interface** CasualtySIf thus may be as follows. It specifies a method getPulseRate(), which must not return a negative value, indicated by the following postcondition: */*@ *ensures* (\ *result* >= 0)@*/*[3]

The same postcondition is specified for the two methods dealing with blood pressure: getSystolicBloodPressure() and getDiastolicBloodPressure(). Moreover an invariant may state the medical knowledge, that the systolic blood pressure must be greater or equal than the diastolic blood pressure: */*@ *public invariant getSystolicBloodPressure() >= getDiastolicBloodPressure(); @*/.

Next to this, a medic application may use **interface** CasualtySIf to set the triage class by executing the method **public void** setTriageClass(TriageClass tc). This enables medic applications in our scenario to calculate the triage class based on the vital data queried from the casualty and set it directly at the casualty. Thus in the following a context-aware medic application can show for example only casualties which are seriously injured.

Imagine medic M2 from Figure 4 is equipped with a medic application, which calculates the triage class in a way, that it sets the triage class to Dead whenever a pulse rate of zero is returned by the casualty. It does not consider the blood pressure in addition. This is correct due to the medic application's implicit assumption that a blood pressure of zero is implied in this case as well. This assumption might have been derived directly from the service interface specification, if this is specified as follows: (getPulseRate()==0) <==> (getSystolicBloodPressure() ==0) <==> (getDiastolicBloodPressure()==0).

However the case might occur, that this assumption will not hold at runtime, since

- the vendor of the casualty service has implemented the service interface incorrectly (*Incorrect Service*)
- the domain architecture does not contain the invariant specified before and thus enables different implementations by different vendors (*Underspecification*).

In practice binding M2's medic application to a casualty, where this assumption does not hold might lead to misclassifications. Let's consider that casualty c's fingerclip measuring the pulse rate slips off. In this case casualty c would be classified as dead by M2's medic application although he still has a blood pressure greater than zero. This would lead to a situation, where no medic is sent to this casualty anymore. Thus it is crucial to detect this incompatibility between medic applications and casualties.

Now assume that due to dependability reasons dynamic checkers like the runtime assertion checker JMLRAC [17] are used during runtime. JMLRAC can be used to execute Java bytecode, which has been compiled by the JML runtime assertion checker compiler jmlc. This bytecode contains specified pre- and postconditions as well as invariants. JMLRAC checks during execution, whether these conditions are satisfied. If any condition is violated, it generates an exception containing, which condition has been violated. In the situation depicted above an exception will state, that an invariant of the casualty service has been violated: the blood pressure is not zero although the pulse rate is zero.

[3] *result* is a JML notation referring to the return value of the specified method.

If we consider medic M1, equipped with a medic application which considers blood pressure as well during triage class calculation, we will recognize another shortcoming. The medic application of M1 sets the triage class to Dead whenever pulse rate as well as blood pressure are equal to zero. This means, that no problem would appear, if this application interacts with the casualty service – even if the fingerclip slips off. Anyhow JMLRAC would state that the invariant of the casualty service has been violated although this would not cause any problems for a binding to this specific medic application.

As you can see, specifications in JML guiding a runtime assertion checker help us to detect the incompatibilities within our example. However we are not satisfied with the results due to major drawbacks: We may detect incompatibilities which are not relevant for the binding and we cannot detect incompatibilities in advance. Service users need to call a method of a service provider in order to realize, that it does not satisfy their assumptions. Therefore we can only detect wrong behavior at the time when it occurs. If you think of an in-car scenario, this corresponds to a situation, where we realize, that the inflateAirbag() method does not work as we expected exactly when we try to inflate the airbag due to a major accident. Thus it does not really help us to establish dependable service bindings. Instead we need to take all possible means to detect incompatibilities in advance.

To address these drawbacks, we provide an approach which enables us to automatically establish service bindings in a system using runtime tests to detect incompatibilities among service users and service providers. However we can only take advantage of such an approach, if we have a service orchestration infrastructure, which uses the test results during service binding. In the past, we developed the Dynamic Adaptive System Infrastructure DAiSI [6] aiming at dynamic adaptive service-based systems.

As motivated before, we want to detect incompatibilities at runtime in advance within this infrastructure. This is done by runtime testing in our approach. The basic idea in our approach is, that service users specify test cases which are executed on a service provider at runtime.

Tests need to be executed by DAiSI before a service user is bound to a service provider (i.e. at binding-time). These tests decide, whether the behavior of a service provider corresponds to the expected behavior defined by a service user in terms of a test case. In our example, the vendor of a medic application might specify such a test case for required casualty services. The test case simply queries the sensor and checks, whether the results are in the expected range. If the tests pass, the integration infrastructure can bind a tested service to its specific service user.

After binding, the compatibility of service provider and service user needs to be monitored, since this may change over time when the internal state of service provider or service user changes. Considering our application example, it does not help, if the tests at binding time pass, since an incompatibility may suddenly come up, when the fingerclip measuring the pulse rate slips off. Thus we need to provide a mechanism enabling us to execute test cases triggered by state changes of the bound service users and service providers.

Our approach here is, defining equivalence classes regarding the state of the service binding. By equivalence classes we understand state spaces of service user and service provider, where the same behavior should apply. If we have these classes, runtime-compliance tests need to be executed each time, when the state changes in a way, that the equivalence class changes as well.

The service provider can define equivalence classes based on the control flow from its implementation. The service user can

 ISBN: 978-1-61208-156-4

define equivalence classes based on its requirements regarding a service provider. Both options alone do not help us: equivalence classes defined for a service binding by a service user can be very different from the ones defined by a service provider.

One reason for incompatibilities that we want to detect is, that the understanding of the service binding and therefore the resulting equivalence class definitions differ. Therefore areas, where the definitions of equivalence classes differ are especially important. They could be missed, if we only define equivalence classes at one endpoint of the binding relation.

Instead our approach is, that equivalence classes for a service binding are defined by both: service users and service providers. This however leads us to the question, how they can be combined in order to derive a changed equivalence class for a service binding. Our approach is very simple: The service provider as well as the service user represent the equivalence class for a service independently simply as a number (e.g. an enumeration of different behavior expectations regarding the service binding).

The equivalence class for the service binding now is the tuple containing the service user's view of the equivalence class and the service provider's view of the equivalence class. We call this tuple a *combined equivalence class*. Whenever a combined equivalence class changes towards an untested combination, runtime-compliance tests need to be executed, since this means, that the behavior of the provided service or the expectations of the service user have changed. The service user needs to associate a specific test case with each of its equivalence classes. The specific test case associated with the current service user's view of the equivalence class is executed, if the combined equivalence class changes.

Looking at our approach from a more abstract point of view, the system executes a sequence as depicted in Figure 5 to determine, whether a specific service provider behaves as expected by a specific service user.

3) Evaluation: For our example this means, that the vendor of the medic application and the vendor of the casualty, define equivalence classes for the used respectively provided service.

The vendor of the casualty may for example decide, that its service provides the same behavior regardless of the internal state. Therefore he implements the getEquivalenceClass () method for both sensors in a trivial way: The provider's view of the equivalence class is 0 all the time (cf. Listing 1).

```
1  public int getEquivalenceClass(){
2    return 0;
3  }
```

Listing 1. A Trivial Implementation of getEquivalenceClass() for a Casualty.

The vendor of M2's medic application associates the equivalence classes of the required casualty service directly with the triage class it calculates for the casualty. The getter for the equivalence classes is depicted in Listing 2. The postfix _casualty in the method name specifies the required service, associated with this equivalence class definition.

```
1  public int getEquivalenceClass_casualty
       (){
2    return calculateTriageClass(casualty);
3  }
```

Listing 2. Implementation of getEquivalenceClass() of the Medic Application for Required Casualties.

Next to equivalence classes, the vendor of M2's medic application needs to define test cases, which are executed by the orchestration infrastructure, whenever the state of service provider or service user changes in a way, leading to an untested combined equivalence class. The execution of these test cases should determine, whether service user and service provider are compatible regarding this service binding in the current state space. Therefore the medic application contains a test case, which could be specified in a specification language like UML Testing Profile [18], [19] or TTCN-3 [20], [21] as depicted in Listing 3.

This test case states, that the medic application in general is not compatible to casualties, when the pulse is out of range or when the triage class is calculated to dead even though one of the sensor values is different from zero. In these cases the test fails, whereas it passes in all other cases[4].

```
1  testcase IsCasualtyCompatible() runs on
       Medic {
2    casualty.call (getPulseRate());
3    casualty.getreply (getPulseRate:{} value
         ?) -> value pulseRate {}
4    casualty.call (getSystolicBloodPressure
       ());
5    casualty.getreply (getPulseRate:{} value
         ?) -> value systolicBloodPressure
       {}
6    casualty.call (getDiastolicBloodPressure
       ());
7    casualty.getreply (getPulseRate:{} value
         ?) -> value diastolicBloodPressure
       {}
8    alt {
9      [] getEquivalenceClass_casualty() ==
           TriageClass.Dead {
10       if (pulseRate != 0 ||
             systolicBloodPressure != 0 ||
             diastolicBloodPressure != 0) {
11         setverdict (fail);
12       } else {
13         if (0<pulseRate<300 && 0<
               systolicBloodPressure<300 &&
               0<diastolicBloodPressure<300)
               {
14           setverdict (pass);
15         } else {
16           setverdict (fail);
17         }
18       }
19     }
20     [] getEquivalenceClass_casualty() ==
           TriageClass.SeriouslyInjured {
21       [...]
22     }
23     [...]
24   }
25 }
```

Listing 3. Testcase for Casualties as Defined by the Medic Application in TTCN-3.

If you now take a look at the situation from our example, we may initially face the situation, that pulse rate as well as blood pressure are measured by the sensors. In this case, the combined

[4]Note, that the test case in our example is very simple in order to focus on the general principles of our approach. Of course we can specify much more complicated test cases using our approach.

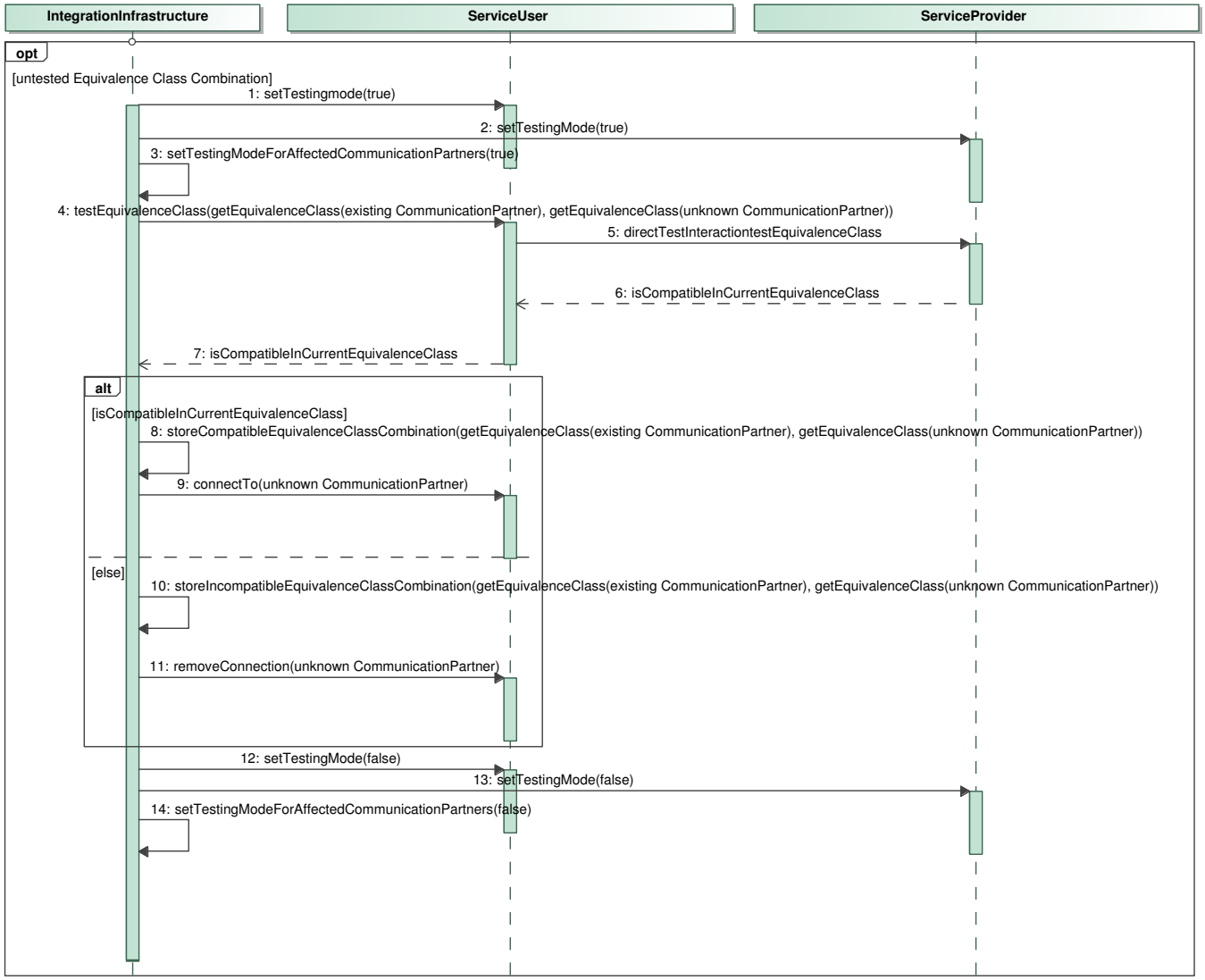


Figure 5. Sequence of the Runtime-Compliance Test in Our Approach.

equivalence class *(Provider: 0, User: 1)* will be calculated at runtime for the service binding between medic application and casualty. The test case defined above will query the sensor values and will pass. In case the fingerclip of the pulse rate sensor slips off, the combined equivalence class changes towards *(Provider: 0, User: 0)* triggering another runtime compliance test. The test case execution fails, since the triage class is calculated to dead although the blood pressure is still above zero. Therefore the orchestration infrastructure will remove the service binding between M2's medic application and the casualty service of casualty c – there might also exist a fallback mode, where the triage class can be manually set by M2.

The test case of M1's medic application would not fail, since it does not require that pulse rate and blood pressure need to be equal to zero at the same time. Thus the orchestration infrastructure would maintain the service binding between M1's medic application and the casualty service of casualty c.

As described above, we are able to detect incompatibilities in advance. However there is still a major drawback of the runtime-testing approach: Since we are testing the interaction between service users and service providers at runtime, we need to ensure that test case execution has no side-effects on our running system. Our approach therefore integrates a so-called testing mode.

Before runtime tests are executed, all involved service providers and service users[5] are notified and can transition into testing mode. Service users respectively providers in testing mode know, that they cannot rely on the interaction with other service providers until this mode is deactivated after test execution. This enables service users respectively providers to restore their state after test execution and to simulate effects (consider the airbag example sketched before: you do not want, that the airbag is inflated due to a test execution, therefore the code inflating the airbag must be substituted by simulation code in

[5]A service user respectively provider is involved, if it is directly or transitively connected by service bindings to the service user respectively provider, which drives the test or which is currently under test

 ISBN: 978-1-61208-156-4

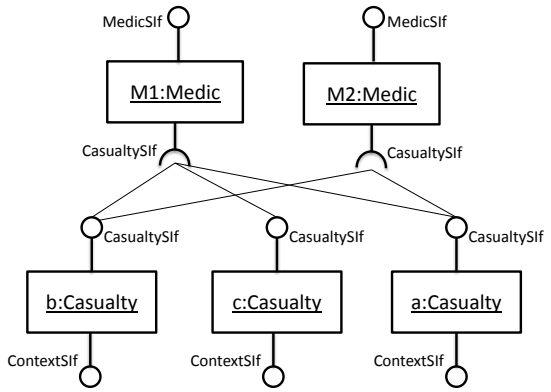


Figure 6. Considered Situation After the Check of Syntactical and Semantical Compatibility.

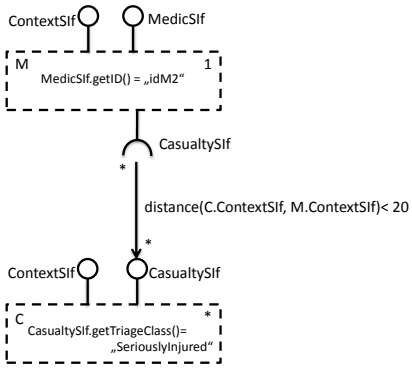


Figure 7. Two Placeholders Defining the Two Required Sets of Services.

testing mode).

Moreover we need to consider, that test-cases may be erroneous. This risk can be limited, if developers of service users use the same test-cases already during development to test mock-ups representing the third-party service providers. This enables the identification of erroneous test-cases at development time.

The runtime testing approach has been implemented within our DAiSI orchestration infrastructure. It enables us to detect incompatibilities and remove incorrect bindings at runtime. The scenario has been exhibited at CeBIT 2009. If you are interested in the application example, you can find a more detailed description at the exhibit's webpage [22]. A more detailed description of our approach and its reference implementation within DAiSI can be found in [23].

B. Context Awareness

In the section before we introduced a method for checking the syntactical and semantical compatibility of services. Another challenge is to use this information to bind those compatible services to useful *context-aware applications*. One solution for the aforementioned challenge is described in this section.

1) Challenge: After restricting the set of services and bindings with respect to syntactical and semantical compatibility, context information has to be considered, in order to bind the services to useful applications. A cutout of the situation after eliminating incompatible services is depicted in Figure 6.

We consider three service providers (the casualties a, b and c), and two service users (the medics M1 and M2) as shown in Figure 4. The depicted bindings between service providers and service users mean that they are syntactically and semantically compatible. Thus, the shown situation is a result of our runtime testing approach introduced in the previous section.

This binding is not useful in all applications respectively in all situations. A useful application in our scenario is to show only those casualties in the medic application which are located close to a medic and which are seriously injured. In Figure 4 you can see that casualty b is located out of range of medic M2 and that casualty e is only slightly injured. Therefore, the only valid binding regarding context awareness is to bind M2 to casualty a, as a is seriously injured and within range of M2. To constraint the service bindings on application level is the challenge we focus on in the following.

2) Approach: As indicated before, additional information about services is needed in order to refine the service binding. To get this information, we use the methods defined in the according service interfaces shown in Figure 3. Additionally we define a service interface called ContextSIf depicted in Listing 4 containing context information relevant for service binding. In our example this interface contains positioning information.

```
1  package de.cadaisi.repository.service;
2
3  import java.awt.Point;
4
5  public interface ContextSIf {
6    public Point getPosition();
7  }
```

Listing 4. The ContextSIf service interface.

We will now have a look at how to choose relevant services for a specific application considering this information first. Later we will introduce how the distance between medic and casualties can be taken into account, too. First we want to define that only those casualties should be connected to the medic which are injured seriously. To do this, we first define two sets of services, one which realizes the medic functionality, and the other realizing the casualty functionality. Figure 7 shows a graphical representation of the two sets of services.

Each dashed rectangle represents a set of services with common properties. We call these rectangles *placeholder*. During runtime these placeholders will be filled with services which meet specific constraints. In the upper-left corner of the rectangles an identifier for each placeholder is given, and in the upper-right corner the required cardinality.

The set of appropriate services is defined by several constraints. First it is given, that a service in placeholder M has to implement the MedicSIf and ContextSIf service interfaces and that it should require the CasualtySIf service interface. Furthermore, a call of the method MedicSIf.getID() should return a value equal to 'idM2'. In that way we pick a specific medic (M2) for the application. For the placeholder C we define similar requirements. A service placed here has to implement the CasualtySIf and ContextSIf service interface and the triage class has to be 'SeriouslyInjured'. Thus, only casualties remain which are seriously injured.

Because properties of services may change over time, the services filling the placeholders also change periodically. Our infrastructure is able to observe these properties during runtime and updates the sets of services accordingly. We call a placeholder activatable, if the number of required services, defined by the cardinality in the upper-right corner of each placeholder, is available.

 ISBN: 978-1-61208-156-4

Having defined these two sets of services, their bindings have to be defined next. In Figure 7 you can see a binding between M and C. In this case we define that each service in M can be bound to an arbitrary number of services in C and that each service in C can be bound to an arbitrary number of services in M. Because we already restricted the set M to exactly one service, the resulting application consists of one medic which is bound to an arbitrary number of casualties.

Furthermore, the binding defines one additional restriction, that is, the maximum distance between services in M and C. We call a binding activatable, if all defined restrictions are fulfilled by a binding. Our middleware considers these bindings as first class entities. It observes their properties and compares them with the according requirements during runtime.

The resulting application is runnable, if all defined placeholders and bindings are activatable. Again, our infrastructure automatically establishes the required bindings as soon as the user starts the application and the application is runnable. In the next section we describe our realization of this infrastructure in more detail.

3) Evaluation: Our infrastructure is able to handle the restrictions described before while considering the syntactic and semantic compatibility of the services. To do this, our developed infrastructure reads an XML based application descriptor where all constraints can be specified by the application developer. Such a descriptor looks like depicted in Listing 5.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <application>
4    <placeholder name="M" cardinality="1">
5      <provides name="MedicSIf">
6        <constraint property="getID()" value
             ="idM2" comparator="de.cadaisi.
             Comparator.equal" />
7      </provides>
8      <provides name="ContextSIf" />
9      <requires name="CasualtySIf" />
10   </placeholder>
11
12   <placeholder name="C" type="Casualty">
13     <provides name="CasualtySIf">
14       <constraint property="getTriageClass
             ()" value="SeriouslyInjured"
             comparator="de.cadaisi.
             Comparator.equal"/>
15     </provides>
16     <provides name="ContextSIf" />
17   </placeholder>
18
19   <binding service="CasualtySIf">
20     <source name="M" cardinality="*" />
21     <target name="C" cardinality="*" />
22     <constraint property="de.cadaisi.
           Context.getDistance(M, C)" value="
           20" comparator="de.cadaisi.
           Comparator.lessThan" />
23   </binding>
24 </application>
```

Listing 5. An Application Descriptor for the Emergency Management System.

In this XML file, two placeholders are defined which represent the placeholders described in the previous section. One requirement for services to fill placeholder M is that they have to implement CasualtySIf. Furthermore, a constraint is given which says that only services can fill this placeholder that return 'idM2' if calling the getID() method. A constraint consists of three parts: a property, a comparator, and a comparative value. Our infrastructure will call this method periodically on services of consideration and check whether the return value meets the condition.

We implemented some predefined comparators in our middleware, but the set of comparators can be enhanced by application developers in order to provide comparison of more complex objects. Finally, the required cardinality for this placeholder to become activatable is given. The placeholder C is defined similarly to M.

Additionally, one binding between these two placeholders is defined within the application descriptor. Here, first the service interface is given which should be bound. The source and target tag define the service user and the service provider, respectively. And finally, it is defined that the distance between service user and service provider should not be larger than 20. Our infrastructure calls the given method getDistance() and compares the return value with the given value, again using the predefined comparator class.

V. Conclusions

Binding service users to service providers in dynamic adaptive service-based systems at runtime is a hard task. Due to the diversity of service users and services, we usually have several options, how we may bind a specific dynamic adaptive service-based system from them. Dependability and specific usage situations help us, to filter meaningful bindings from these options. Thus we can derive meaningful service bindings by applying some simple mechanisms. As we did not encounter performance problems within our application examples, we did not evaluate the performance impact of our approach, yet.

We discussed two aspects of service bindings: semantical compatibility and context awareness. To identify semantical incompatible service bindings, we proposed a runtime testing approach. It is based on test cases defined by a service user.

The semantical compatibility of service bindings may change during runtime as the internal state of a service provider respectively service user. Since runtime testing only gives a snapshot of semantical compatibility at the time of test case execution, we propose to define equivalence classes for a service binding. An equivalence class states, that equivalent behavior of the associated service is provided respectively expected.

These equivalence classes are defined by the service users as well as at the service providers side. An orchestration infrastructure can monitor the equivalence classes and repeat test execution when one of those equivalence classes changes at runtime. Thus we can identify service bindings which are semantical incompatible and prevent that they are established at system runtime by our orchestration infrastructure.

The second aspect we discussed in this article was the context-aware service binding. Based on the identified syntactical and semantical compatible service bindings, a useful application has to be built which is able to react on context changes. To do this, we introduced the placeholder concept, where each placeholder defines a set of services based on common characteristics. These characteristics are on the one hand provided and required service interfaces. On the other hand we showed how we make use of information provided by service interfaces in order to refine the according set of services. Additionally, possible bindings between services have been restricted also using information provided by service interfaces.

 ISBN: 978-1-61208-156-4

Based on our existing infrastructure DAiSI [6] we realized an orchestration infrastructure, which is capable of deriving and establishing valid and meaningful service bindings regarding dependability [23] and context awareness. Using DAiSI we can establish service bindings in dynamic adaptive service-based systems automatically at runtime. Thus a vendor of a service-oriented application does not need to deal with service discovery and binding anymore, since this task is performed by our infrastructure.

An open issue regarding service binding is the involvement of the user. Despite considering dependability and context awareness, our orchestration infrastructure may determine multiple valid service bindings. Our infrastructure cannot reason about the quality of these bindings any further and thus will establish any of these bindings. However the user might be able to chose a service binding based on his specific goals. One of these situations occurs in our scenario; the medic M3 is injured and has to decide if he either wants to be displayed as a casualty or as a medic. Since the infrastructure cannot decide this, it makes a user-driven decision indispensable. This user-integration into the binding process is only at a conceptual stage at the moment.

ACKNOWLEDGMENT

The runtime testing approach for dependable service bindings presented here has been elaborated in cooperation with Cornel Klein, Jürgen Reichmann, and Reiner Schmid from Siemens AG. Together we filed a patent for it.

This work was partly funded by the NTH School for IT Ecosystems. NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover. Furthermore, the work was partly funded by OPEN (Open Pervasive Environments for migratory iNteractive services), an VII Framework EU STREP project.

Many thanks to the reviewers for their helpful comments enabling us to improve this paper.

REFERENCES

[1] O. Alliance, *OSGi Service Platform Core Specification*, 2007.

[2] SOPERA, "Sopera enterprise service bus," http://www.sopera.com [Online; accessed 17-November-2009].

[3] Microsoft, "Managed extensibility framework," http://www.codeplex.com/MEF [Online; accessed 17-November-2009].

[4] K. Herrmann, K. Geihs, and G. Mühl, "Ad hoc service grid - A self-organizing infrastructure for mobile commerce," in *Proceedings of the IFIP TC8 Working Conference on Mobile Information Systems (MOBIS 2004)*. Oslo, Norway: Kluwer, Sep. 2004, pp. 261–274.

[5] Triskell Project, "Kermeta," http://www.kermeta.org [Online; accessed 17-November-2009].

[6] D. Niebuhr, H. Klus, M. Anastasopoulos, J. Koch, O. Weiß, and A. Rausch, "DAiSI - Dynamic Adaptive System Infrastructure," Fraunhofer Institut Experimentelles Software Engineering, IESE-Report No. 051.07/E, Tech. Rep., Jun 2007.

[7] Manchester Triage Group, *Emergency Triage*. BMJ Books, 2005.

[8] R. B. Smith, B. Horan, J. Daniels, and D. Cleal, "Programming the world with sun spots," in *OOPSLA '06: Companion to the 21st ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. New York, NY, USA: ACM, 2006, pp. 706–707.

[9] A. Rausch, "Software evolution in componentware using requirements/assurances contracts," in *ICSE 22, Proceedings of the 22th International Conference on Software Engineering*, Jun 2000.

[10] A. M. Turing, "On computable numbers, with an application to the entscheidungsproblem," in *Proceedings of the London Mathematical Society*, 1936, pp. 230–265.

[11] A. Church, "An unsolvable problem of elementary number theory," *American Journal of Mathematics*, vol. 58, pp. 345–363, 1936.

[12] B. Zimmerova, P. Vařeková, N. Beneš, I. Černá, L. Brim, and J. Sochor, "Component-interaction automata approach (coin)," pp. 146–176, 2008.

[13] A. Both and W. Zimmermann, "Automatic protocol conformance checking of recursive and parallel component-based systems," in *CBSE*, 2008, pp. 163–179.

[14] E. Estévez and P. R. Fillottrani, "Bisimulation for component-based development," *Journal of Computer Science & Technology*, vol. 1, no. 6, May 2002.

[15] B. Meyer, "Applying "design by contract"," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[16] G. T. Leavens, A. L. Baker, and C. Ruby, "Preliminary design of jml: a behavioral interface specification language for java," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 3, pp. 1–38, 2006.

[17] L. Burdy, Y. Cheon, D. Cok, M. Ernst, J. Kiniry, G. Leavens, K. Leino, and E. Poll, *An overview of JML tools and applications*, 2003.

[18] UML Testing Profile Webpage, http://tinyurl.com/yhxrhlp [Online; accessed 29-November-2007].

[19] P. Baker, Z. R. Dai, J. Grabowski, O. Haugen, I. Schieferdecker, and C. Williams, *Model-Driven Testing: Using the UML Testing Profile*, 1st ed. Springer, Berlin, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-72563-3

[20] Testing & Test Control Notation Webpage, http://www.ttcn-3.org/ [Online; accessed 29-November-2007].

[21] C. Willcock, T. Dei, S. Tobies, S. Schulz, S. Keil, and F. Engler, *An Introduction to TTCN-3.*, 1st ed. Wiley & Sons, Apr. 2005.

[22] D. Niebuhr, M. Schindler, and D. Herrling, "Emergency assistance system – webpage of the cebit exhibit 2009," http://www2.in.tu-clausthal.de/%7ERettungsassistenzsystem/en [Online; accessed 09-February-2009].

[23] D. Niebuhr, "Dependable Dynamic Adaptive Systems – Approach, Model, and Infrastructure," Ph.D. dissertation, Technische Universität Clausthal, 2010.

The Role of Corticothalamic Feedback in the Response Mode Transition of Thalamus

Jia-xin Cui
Institute of Psychology
Chinese Academy of Sciences
Beijing, China
e-mail: cuijiaxin@gmail.com

Chun-feng Shang
Institute of Neuroscience
Chinese Academy of Sciences
Shanghai, China
e-mail: cfshang@ion.ac.cn

Abstract—Thalamus is the relay station for most sensory information between peripheral receptors and the cortical areas. There is cumulative evidence showing that thalamus actively gates information flow. An important issue about this role is that thalamus adjusts its response mode according to the sensory information arrived. The corticothalamic feedback signal has been suggested to be essential for such adjustment. In this paper, we pieced together the experimental evidence in a realistic multi-layer network model of the thalamocortical circuit and examined its behavior in response to sinusoidal inputs. The results reproduced the burst/tonic responsive modes and feedback signal's role in mode transition.

Keywords-thalamus; cortex; feedback; response mode.

I. INTRODUCTION

While sensory signals are detected in peripheral receptors, cortical processing of such signals is responsible for the perceptual experience. On the passage from periphery to cortex, there is the relay station of thalamus. Cumulating evidence has shown that the relay function of thalamus is not simply "passing by" but active processing. This processing is modulated by various inputs to thalamus other than those from the peripheral sensory organs. These inputs, including those from cerebral cortex and brainstem, modulate the thalamic nuclei's state and control their gating properties in neuronal information processing.

Since 1990s, the thalamic relay cells have been shown to exhibit two distinct modes, burst and tonic, in responding to the sensory inputs [3]. Neurons in burst mode are shown to be more efficient in signal detection and found mostly when the animal is in drowsy or sleeping state. In contrast, in tonic mode neurons have higher fidelity in signal transmission and tonic mode is related to alert and behaving states. The transition between burst and tonic modes is suggested to be dependent on the recent membrane potential history before the coming of sensory information. Based on such results and the reciprocal connection between thalamic relay nuclei and their related cortical areas, there emerges the "wake-up call" hypothesis that the feedback information from cerebral cortex can switch the relay cells from burst mode to tonic mode by depolarization. This hypothesis is attracting since it offers a versatile transfer function dealing with various stimuli and brain states like a self-adapting filter. The evidence supporting this hypothesis is still being gathered and in need of tidying up. Murray Sherman and his colleagues have illustrated the kinetics of low threshold calcium channel (I_T) [4] and shown that it may contribute to the transition between burst and tonic modes of thalamic relay cells in an integrate-and-fire-or-burst model [5].

Based on their work, we want to test the "wake-up call" hypothesis in a multi-layered network with integrate-and-fire-or-burst neurons. The model is focused on the interaction between thalamus and cortex. Neocortex is composed of 6 layers in which layer IV and layer VI receive thalamic inputs. While the signal to layer IV are passed through layer II/III, layer V, and then to higher thalamic nuclei and brainstem, layer VI sends feedback afferent to the thalamic relay nuclei from which it receives input. Given this specific connection of layer VI and our simulation resource limit, we omitted the cortical layers other than layer VI and higher thalamic nuclei from our current model. Our results showed that feedback is essential in controlling the relay cells' mode transition. We also explored the input frequency's role in shaping the stimulation-response mode relationship and proposed that this supports that the mode transition is essential in controlling the animal states.

In the contribution below, we first introduce the network structure of our model, then list parameters employed in the simulation, and finally show simulation results about the network under various stimulation paradigm. The significance of the results is discussed finally.

II. METHODS

A. Network structure

The network comprises two layers, one thalamic and one cortical. There were five populations in thalamic layer, resembling cells in a barrel-like structure of relay nucleus, with various tuning on stimulation features, such as directions and velocity of whisker reflection. Each relay cell population (resting potential, V_{relay}) was reciprocally and specifically connected with a neuron population in the cortical layer. The reciprocal connection was assigned according to the anatomical features of layer VI excitatory neurons (resting potential, V_{pyr}), which offer the corticofugal feedback signals [6]. The specific one-to-one connection was assigned since the stimulation features are preserved in the thalamocortical projections. There was a representative homogenous population of layer VI inhibitory interneuron, basket cell (resting potential, V_{bask}), in the cortical layer. According to Swadlow et al [7], the thalamocortical inputs to

 ISBN: 978-1-61208-156-4

the layer IV interneurons are divergent and convergent, blurring the stimulation features. Similarly we assigned layer VI inhibitory interneuron with inputs from all thalamic relay cells and outputs to all cortical excitatory cells, sharing the same number with thalamic cells. The 1 from 6 ratio of inhibitory interneurons is near the general proportion, 20~30 % in cerebral cortex [8].

B. The integrate-and-fire-or-burst model

The neurons in our network were described by the integrate-and-fire-or-burst (IFB) model taken from [5] and [9]. The IFB model was constructed by adding a slow variable to a classical integrate-and-fire neuron model. The neuronal model contains two equations, one for voltage and one for calcium current:

$$C \times dV/dt = -I_L - I_T - I_S - I_D \quad (1)$$

$$dh/dt = -h/\tau_h^-, \text{ if } V \geqslant V_h;$$

$$(1-h)/\tau_h^+, \text{ if } V < V_h \quad (2)$$

where $I_L = g_L \times (V - V_L)$ is a constant conductance leakage current, I_T is the low threshold Ca^{2+} current, and I_S and I_D are two synaptic currents. An action potential occurred whenever the membrane potential reaches the firing threshold (V_{theta}). There was an absolute refractory period τ_{rp} (4 ms) during which the voltage of neurons remains at V_{reset}. The low-threshold Ca^{2+} current was described as $I_T = g_T \times m_\infty \times h \times (V - V_T)$, where $m_\infty = \Theta(V - V_h)$ represents instantaneous voltage-dependent activation, and Θ is the Heaviside step function.

The input to the network was as following: (1) simulation begins with a pre-cue time interval lasting for 500 ms, during which the network receives only noise and will exhibit spontaneous activity; (2) in the stimulus presentation period, the input of sinusoidal current $I_0 + I_I * \sin(f)$ lasted for 5000 ms was applied to the thalamic cells besides for noises from outside.

C. Synaptic input

The last two terms in (1), I_S and I_D, were synaptic currents attributable to excitatory spontaneous input and excitatory or inhibitory drive. Each synaptic potential received by the neuron in the network was modeled as an α function [10] of conductance. An individual EPSP occurring at t =0 would be given by:

$$\alpha(s) = q/\tau_s \times \exp(-s/\tau_s) \times \Theta(s) \quad (3)$$

where q is the total charge that is injected in a postsynaptic neuron via a synapse with efficacy $w_{ij} = 1$. More realistically, the postsynaptic current α should have a finite duration, e.g., as in the case of an exponential decay with time constant τ_s. As usual, Θ is the Heaviside step function with $\Theta(s) = 1$ for $s > 0$ and $\Theta(s) = 0$ else.

All calculations were performed using the Matlab program and a time step of 0.1 ms.

D. Cellular and synaptic parameters

These parameters were collected from several sources and listed below, as table 1, 2, and 3.

III. RESULTS

A. Burst and tonic modes in relay cells

We thoroughly tested the model's response to several sets of parameters including baseline current, amplitude of sinusoidal inputs and sinusoidal frequency. Two representative results are showed in Figure 1a and 1b. The baseline current strongly determines the response mode of relay cells, consistent with Sherman et al's results that the low and high baseline current lead to burst and tonic mode, respectively.

B. Frequency dependency of mode transition

The whole sets of results are shown in Figure 2. For both low baseline situations with (Figure 2a) and without feedback (Figure 2b), the response mode transition is dependent on input frequency. At both situations, the emergence of burst mode response from originally silent cells are relatively transiently as the stimulus increases for the high frequency stimulation and smoothly for the low

TABLE I. CELLULAR PARAMETERS

Para-meters	Values and Sources		
	Value	Unit	Reference
V_θ	-35	mV	[5]
V_L	-65	mV	[5]
V_{reset}	-50	mV	[5]
C	2	$\mu F*cm^{-2}$	[5]
g_L	0.035	$mS*cm^{-2}$	[5]
V_h	-60	mV	[5]
V_T	120	mV	[5]
τ_h^-	20	Ms	[5]
τ_h^+	100	Ms	[5]
g_T	0.07	$mS*cm^{-2}$	[5]
V_{relay}	-65±3.44	mV	[11]
V_{pyr}	-67	mV	[12]
V_{bask}	-62	mV	[8]

TABLE II. SYNAPTIC PARAMETERS

Presynaptic	Postsynaptic	Values and References			
		Synapse Number	Strength ($\mu A*cm^{-2}$)	Synaptic delay(ms)	Time constant (ms)
Relay cells	Pyramidal cells	20 [13]	3.5 [14]	1 [15]	2 [13]
Relay cells	Basket cells	20 [13]	5 [15]	1 [16]	1 [13]

 ISBN: 978-1-61208-156-4

Presynaptic	Postsynaptic	Values and References			
		Synapse Number	Strength ($\mu A*cm^{-2}$)	Synaptic delay(ms)	Time constant (ms)
Basket cells	Pyramidal cells	20±1.34 [13]	1.4 [14]	0 [13]	6 [17]
Pyramidal cells	Relay cells	20 [13]	2 [18]	5 [19]	2 [17]

TABLE III. STIMULUS PARAMETERS

Parameters	Values and Units	
	Value	Unit
Baseline I_0	-0.05 / 1.1	$\mu A*cm^{-2}$
Amplitude I_1	0.1~2	$\mu A*cm^{-2}$
Frequency f	0.25 / 1 / 5 / 25	Hz

a. The stimuli are $I = I_0 + I_1\cos(2\pi f)$

frequency stimulation. The transition from burst mode to tonic mode happens at stimuli higher than 1 $\mu A*cm^{-2}$, but not for all stimulation frequencies. Figure 2c is the high baseline situation with or without feedback.

C. Feedback connection shapes the mode Transition

The burst-tonic transition was essential to the "wake-up call", suggesting that the feedback signal was playing its role and the relay cells were switched into the high fidelity transferring mode from the high-efficacy detecting mode. For 5 Hz and 25 Hz stimuli with low baseline, the burst-tonic mode transition happens only if there is the feedback connection. For the 0.25 Hz stimuli with low baseline the

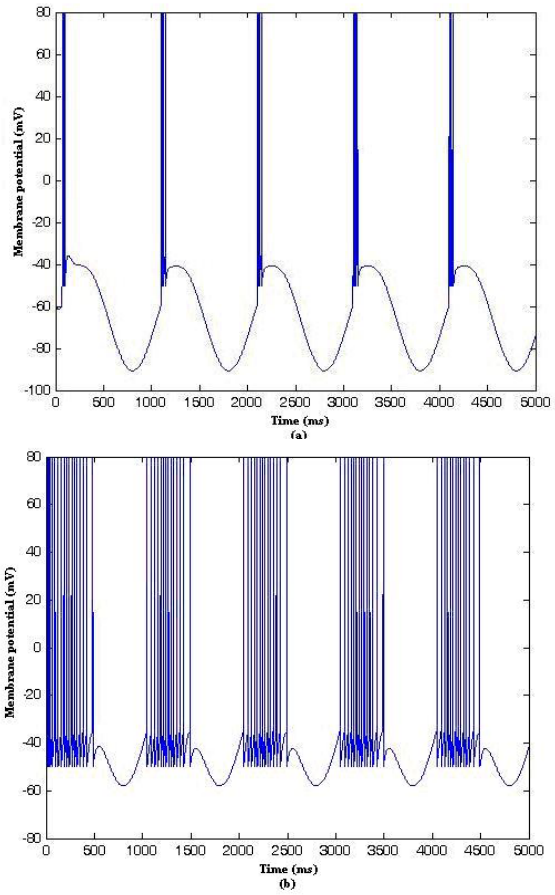
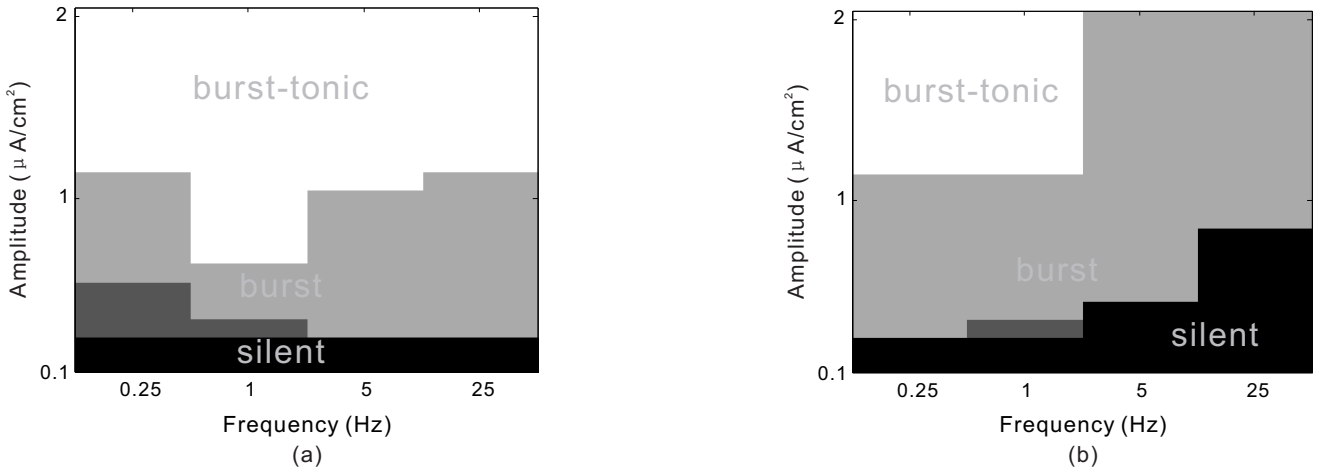


Figure 1. Burst and tonic modes in the relay cells: (a) burst mode, the stimulus is I_0=0.9*cos (π/500) -0.05. (b) tonic mode, the stimulus is I_0=0.9*cos (π/500) +1.1.



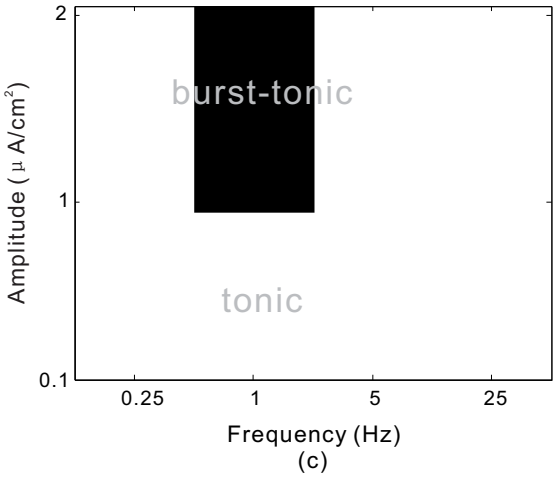


Figure 2. Mode transition. The response modes to various stimuli represented in colorgram. The low baseline situation is sensitive to feedback connections while the high baseline situation is resistant to it. (a, b) the response modes of relay cells with or without feedback connections to low baseline stimuli. (c) the response modes of relay cells to high baseline stimuli.

stimulation-response mode relationship was not strongly dependent on the existence of feedback connection. The high baseline stimuli raised the relay cells already to tonic mode, so the relay cells were mostly resistant to remove of feedback connections. The 1 Hz stimuli with high baseline showed unique feature that the burst-tonic mode transition was evoked when the stimuli was up to 0.9 $\mu A*cm^{-2}$.

D. I_T current is underlying the transition

To verify the essential role of low threshold calcium channel we compared the mode transition graph and the activation states of I_T. Since the activation states of I_T is governed by membrane potential and the slow variable h, and the time constant of h is invariable, we plotted the duration length in which h is positive. $m_\infty = \Theta(V - V_h)$, and Θ (*) is the Heaviside step function. Consistent with the physiological result, the active periods were mostly about 100 ms when the stimuli are capable of evoking burst-tonic transition.

IV. DISCUSSION

Although the model is really simple and incomplete, the essential role of feedback is significant. To incorporate more components, especially the reticular nucleus and cortical columnar constructions should make the model more concrete and more precise in temporal structure. The feedforward inhibition is proposed to be efficient in shaping the receptive field spatially and temporally. Since it is incorporated in the model, it is possible to study that in the multi-layered neuronal network instead of in pure feedforward triad. In fact there has been work on this idea [20].

The stimuli frequencies are related to different brain states: 0.25 Hz are slow waves related with drowsiness and slow-wave sleep, 1 Hz are δ-wave related with slow-wave sleep, 5 Hz are θ-wave related with certain behavioral activities, and 25 Hz are α-wave related with restful attentiveness [21]. The slow-wave and δ-wave are related with similar brain states, and so are the δ-wave and θ-wave. Consistently, stimuli of the former two frequencies evoke similar response mode and show similar response to the existence of feedback connections. Similar comparison applies to the later two. This offers us a new point of view to consider the role of feedback connections. The burst-tonic transition in feedback-removed relay cells to low frequency stimuli may reflect the higher sensitivity to silent signals during the drowsy states.

Because the neuronal model of this work is a single-compartmental model, there is a limitation that some results may not expand to situations involving temporally or spatially patterned synaptic input distributed over dendritic arbors. Nevertheless, this kind of single-compartmental model is appropriate in some situations, and the statistics for the stochastic responses is satisfied. However, a model with multi-compartment may have more fruitful results.

ACKNOWLEDGMENT

This work is supported by the China Postdoctoral Science Foundation funded project under Grant No. 20100470593 and the 973 Program under Grant No. 2011CB302201.

We are very grateful of Santa Fe Institute to offer us the opportunity to study in the Complex System Summer School in Beijing 2005 and make us known. We thank Zhang Jiang and Li Jing-jing for fruitful discussion, prof. John Holland, prof. Han Jing, prof. Cosma Shalizi for valuable suggestions.

REFERENCES

[1] S. Sherman and R. Guillery, "The role of the thalamus in the flow of information to the cortex," Philosophical Transactions of the Royal Society Lond. B, vol. 357, Dec. 2002, pp. 1695-1708.

[2] W. Guido S.-M. Lu. Vaughan, D.Godwin and S.Sherman, "Receiver operating characteristic (ROC) analysis of neurons in the cat's lateral geniculate nucleus during tonic and burst response mode," Visual Neuroscience, vol. 12, Jul-Aug. 1995, pp. 723-741.

[3] S. Sherman, "Tonic and burst firing: dual modes of thalamocortical relay," Trends in Neuroscience, vol. 24, Feb. 2001, pp. 122-126.

[4] X. Zhan, C. Cox, J. Rinzel and S. Sherman, "Current clamp and modeling studies of low threshold calcium spikes in cells of the cat's lateral geniculate nucleus," Journal of Neurophysiology, vol. 81, May. 1999, pp. 2360–2373.

[5] G. Smith, C. Cox, S. Sherman and J. Rinzel, "Fourier analysis of sinusoidally-driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model," Journal of Neurophysiology, vol. 83, Jan. 2000, pp. 588–610.

[6] G. Shepherd, The Synaptic Organization of the Brain. New York: Oxford University Press, 2004

[7] H. Swadlow, "Fast-spike interneurons and feedforward inhibition in awake sensory neocortex," Cerebral Cortex, vol. 13, Jan. 2003, pp. 25-32.

[8] H. Markram, M.Toledo-Rodriguez, Y. Wang, A. Gupta, G. Silberberg and C. Wu, "Interneurons of the neocortical inhibitory system," Nature Reviews Neuroscience, vol. 5, Oct. 2004, pp. 793-807.

 ISBN: 978-1-61208-156-4

[9] G. Smith and S. Sherman, “Detectability of Excitatory versus Inhibitory Drive in an Integrate-and-Fire-or-Burst Thalamocortical Relay Neuron Model,” Journal of Neuroscience, vol. 22, Dec. 2002, pp. 10242–10250.

[10] W. Rall, “Distinguishing theoretical synaptic potentials computed for different soma-dendritic distributions of synaptic inputs,” Journal of Neurophysiology, vol. 30, Jan. 1967, pp. 1138–1168.

[11] H. Jahnsen and R Llinas, “Electrophysiological properties of guinea-pig thalamic neurones: an in vitro study,” Journal of Physiology, vol. 349, Apr. 1984, pp. 205-226.

[12] C. Shang, unpublished data.

[13] R. Traub, D Conteras, M. Cunningham, H. Murray, F. LeBeau, A. Roopun, et al., “Single-column thalamocortical network model exhibiting gammaoscillations,sleep spindles, and epileptogenic bursts,” Journal of Neurophysiology, vol. 93, Apr. 2005, pp. 2194-2232.

[14] N. Bannister, J.Nelson and J.Jack, “Excitatory inputs to spiny cells in layers 4 and 6 of cat striate cortex,” Philosophical Transactions of the Royal Society Lond. B, vol. 357, Dec. 2002, pp. 1793-1808.

[15] A.Agmon and B.Connors, “Correlation between intrinsic firing patterns and thalamocortical synaptic responses of neurons in mouse barrel cortex,” Journal of Neuroscience, vol. 12, Jan. 1992, pp. 319-329.

[16] J. Porter, C. Johnson and A. Agmon, “Diverse Types of Interneurons Generate Thalamus-Evoked Feedforward Inhibition in the Mouse Barrel Cortex,” Journal of Neuroscience, vol. 21, Apr. 2001, pp. 2699-2710.

[17] J. Turner and T. Salt, “Characterization of sensory and corticothalamic excitatory inputs to rat thalamocortical neurones in vitro,” Journal of Psysiology, vol. 510, Aug. 1998, pp. 829-843.

[18] C. Kao and D. Coulter, “Physiology and Pharmacology of corticothalamic stimulation-evoked responses in rat somatosensory thalamic neurons in vitro,” Journal of Neurophysiology, vol. 77, May. 1997, pp. 2661-2676.

[19] L.Gentet and D. Ulrich, “Strong, reliable and precise synaptic connections between thalamic relay cells and neurones of the nucleus reticularis in juvenile rats,” Journal of Physiology, vol. 546, Feb. 2003, pp. 801-811.

[20] Y. Choe, “Processing of analogy in the thlamocortical circuit,” Proc. IEEE Symp. Proceedings of the International Joint Conference on Neural Networks, IEEE Press, May 2003, pp. 1480-1485.

[21] M. Castro-Alamancos and B. Connors, “Thalamocortical synapses,” Progress in Neurobiology, vol. 51, Apr. 1997, pp. 581-606.

 ISBN: 978-1-61208-156-4

Adaptive Mobile Web Applications Through Fine-Grained Progressive Enhancement

Heiko Desruelle, Dieter Blomme, Frank Gielen
Ghent University – IBBT
Dept. of Information Technology – IBCN, Ghent, Belgium
{*heiko.desruelle, dieter.blomme, frank.gielen*}*@intec.ugent.be*

***Abstract*—The availability of mobile devices is growing at an incredible pace. This trend has set a need for applications being available at anytime, anywhere, and on any device. As most mobile users carry their device at all times, being truly mobile provides users an unprecedented freedom. Nevertheless the clear advantages, application developers are facing challenges due to device fragmentation. Current application development solutions are insufficiently prepared for the high variety of mobile platforms and hardware characteristics. In this paper we propose a platform for the development and delivery of adaptive mobile applications. An adaptive application composition approach is introduced, capable of autonomously bypassing fragmentation related issues. This goal is achieved by incorporating fine-grained progressive application enhancements through a quantitative evaluation strategy.**

***Keywords*-mobile web, adaptive mechanism, progressive enhancement, quantitative evaluation.**

I. INTRODUCTION

Mobile is a powerful mass medium, with a greater reach and faster growth than any other known media type [1]. Furthermore, the technology itself is rapidly maturing. Advanced features such as mobile internet access and integrated sensors have become a standard capability of devices throughout all user segments. As users carry their mobile device practically at all times, a need has been established for mobile applications and services being available anywhere and at any time. However, various technological challenges reside in the development of applications that automatically cover many types of devices [6]. This barrier is a result of the heavily fragmented mobile landscape. In order to attain a sustainable share of the mobile market, applications need to be made adaptable to various combinations of hardware, operating systems, APIs, etc. The absence of appropriate platform and tool support make it the developer's responsibility to deal with adaptability requirements, which considerably drives up development costs and narrows target markets. Against this backdrop, the use of the web as an application platform is gaining momentum. Device independent web technologies such as HTML, CSS and JavaScript offer application developers an unprecedented market reach. Furthermore, the International Telecommunication Union (ITU) estimates the use of mobile internet connectivity to surpass access rates of traditional desktop-based internet by the year 2013 [7].

Web systems are traditionally engineered along three orthogonal dimensions: the development phases, the system's views, and its aspects (as illustrated in Figure 1) [11]. The phase dimension sets out the different stages of web development, ranging from analysis to implementation. Each of these phases requires a number of specific views addressing the systems content, its navigation structure, and the presentation. Finally, the aspects dimension defines the structural and behavioral aspects of each of the views. The growing presence of mobile applications emphasizes the need for fragmentation management within the web engineering model. This concern has to be handled throughout every stage of the application's development life cycle. As proposed by Kappel et al., adaptability can be considered as an additional web engineering dimension, crosscutting all other web modeling dimensions [9]. From a developer's perspective, the straightforward incorporation of adaptability remains an important challenge [11]. Even with the use of standardized web technology, efficiently managing mobile fragmentation remains an important research topic. The different mobile browsers still contain many variability points, making true mobile convergence not to be expected any time soon [6].

Since the early days of web engineering, developers have tried to cope with the differences between browsers. Graceful degradation is a widespread design strategy that focuses on providing optimal support for the most advanced browsers. Less capable browsers are only considered during the last development phase. This approach often results in a poor stripped-down version. The graceful degradation

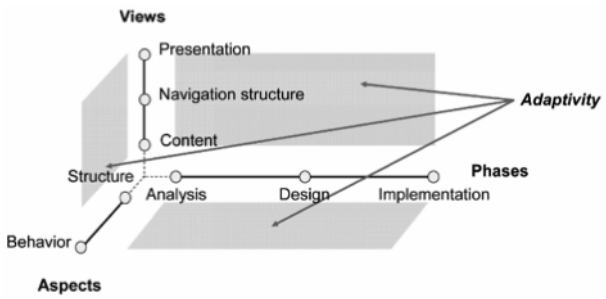


Figure 1. Adaptability as a crosscutting aspect on the traditional modeling dimensions of web engineering (from Koch et al. [10])

 ISBN: 978-1-61208-156-4

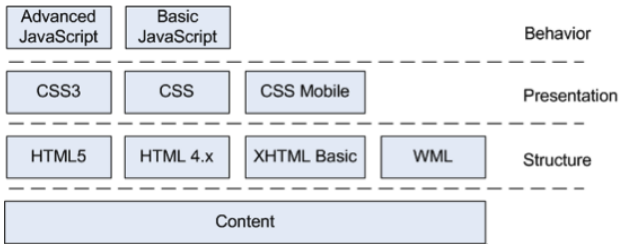


Figure 2. Fine-grained mobile progressive enhancement. A dynamic process, driven by the specific capabilities of a client's mobile device

methodology expects users to just upgrade their browser when the degraded version does not fit their needs. However, for most mobile devices upgrading the default browser is not an option.

Progressive enhancement (PE), on the other hand, reverses the graceful degradation approach and aims at maximizing accessibility over browsers with different capabilities [13]. Progressive enhancement tries to achieve this goal by forcing developers to take the less capable devices into account from the very start of the development process. First, a basic markup document is created, providing an optimal experience for devices with the lowest common denominator (LCD) of available capabilities. Incrementally and unobtrusively, one or more layers of structural, presentational, and behavioral enhancements are added in function of the browser's specific capabilities.

The progressive enhancement methodology can be applied in a mobile context to tackle fragmentation related issues. However, when turning the theoretical approach into actual practice, a considerable number of challenges come into play. Today, the use of CSS3 Media Queries [15] and externally linked resources are the most common practice for selecting appropriate enhancement layers. The number of detectable variability points is limited, as adaptation can only be performed based on the device's screen capabilities and coarse-grained styling and scripting support. Compared to desktop browsers, the mobile ecosystem contains far more combinations of browsers with graded CSS and JavaScript support. To provide optimized end-user usability, progressive enhancement should also reckon with the different interaction methods and hardware characteristics offered by mobile devices. For example, a touch-based device will often require an additional presentational enhancement layer, providing a user interface with more space to accurately click buttons, links, etc.

In order to create a viable progressive enhancement solution, it has become increasingly important to support the use of more fine-grained enhancement layers. As shown in Figure 2, an intelligent mechanism is needed, supporting the automated creation of progressive enhancement stacks based on the specific capabilities of a user's mobile device. Within this context, the goal of our research is to introduce such a mobile progressive enhancement platform. To support developers in the creation of adaptive mobile applications, we propose a method on how to extend existing application frameworks with mobile progressive enhancement capabilities. Furthermore, we introduce an adaptive application composition algorithm, which will be at the heart of composing optimal progressive enhancement stacks. With this approach, we propose a robust and future proof method for the flexible composition of web applications based on the specific capabilities of a user's mobile device.

The remainder of this paper is structured as follows. Section II discusses the algorithmic structure of our approach. Section III deals with the architectural aspects of extending application frameworks with mobile progressive enhancement capabilities. In section IV we discuss the proof of concept implementation of the architecture, followed by some evaluation results. Finally, future work and our conclusion are presented in Section V.

II. Adaptive Application Composition Algorithm

In this section we propose an adaptive application composition algorithm, realizing the above defined objective to enable the capability-driven progressive enhancement of web applications. Mobile applications should provide users with an optimal experience based on the specific capabilities of their device. In order to cope with the wide variety of mobile characteristics, we introduce a quantitative evaluation algorithm derived from the Logic Scoring of Preference (LSP) method. This adaptive application composition algorithm is designed to support fine-grained progressive enhancement and is capable of suggesting a stack of layers that optimally fits the user's mobile device.

A. Basic Algorithm

LSP is a quantitative decision method, proposed by Dujmovidc [3]. It is designed to assist decision makers in the evaluation, comparison, and selection of complex hardware and software systems. The method has shown its use in various domains, especially concerning situations with large and complex solution spaces. To evaluate a set of candidate solutions, LSP starts by assessing n individual performance variables. These variables define the n properties that an ideal solution is expected to have. As the algorithm deals with complex decision problems, most candidate solutions will not perfectly match the preset criteria. Nevertheless, such candidates should not be rejected, as their overall evaluation might still lead to an acceptable solution. LSP addresses this issue by taking into account how well a candidate matches the different performance variables. For each variable i, a degree of suitability $E_i \in [0, 1]$ is calculated. This score expresses the similarity between a candidate solution and performance variable i, ranging from 0 to 100 %. In order to attain these scores, LSP requires a predefined mapping function for each performance variable

 ISBN: 978-1-61208-156-4

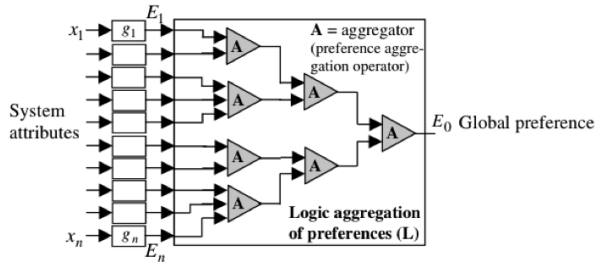


Figure 3. LSP score aggregation in order to derive an overall suitability score from the elementary degrees of similarity (from Dujmovic et al. [4])

[4]. Both Table I and II below illustrate such mapping functions in a mobile context.

After obtaining the elementary degrees of satisfaction, all individual matching scores are to be combined into one objective overall suitability score. This aggregated score is used to determine the best-matching candidate. LSP supports the use of aggregation networks, expressing the mutual relationships between individual scores and how to calculate the overall score (see Figure 3). The standard aggregation operators in LSP are based on the superposition of fundamental Generalized Conjunction Disjunction (GCD) [2]. These operators enable aggregations in terms of partial conjunction, full conjunction, partial disjunction, full disjunction, and neutrality in a single operator. Moreover, a GCD supports the specification of aggregations in terms of 17 graded combinations of conjunction and disjunction. A frequently used implementation for GCD are Weighted Power Means (WPM)

$$WPM\left(x_1, x_2, \ldots, x_m; W_1, W_2, \ldots, W_m; r\right) = \left(W_1 x_1^r + W_2 x_2^r + \ldots + W_m x_m^r\right)^{\frac{1}{r}}. \quad (1)$$

The variables W_i in Equation 1 represent the relative weight for each elementary degree of suitability x_i, where $W_1 + \ldots + W_m = 1$. The exponent r is determined in function of the aggregation's desired degree of cunjunction or disjunction. This approach allows an evaluator to precisely couple the mutual importance of individual suitability degrees. The calculated aggregation network results in an objective overall suitability score

$$E = L\left(E_1, \ldots, E_n\right) \quad (2)$$

where the function L is a combination of one or more GCDs using the individual suitability degrees as input parameters. After calculating E for each of the candidates, conclusions regarding the best-matching solution can be drawn. The LSP approach selects the candidate with the highest overall suitability score as the optimal choice.

B. LSP in a Mobile Context

LSP has the ability to flexibly, yet objectively, evaluate systems under various circumstances. We propose a modification of the LSP method in order to support the adaptive composition of mobile web applications. In this particular case, the stacks of progressive enhancement layers are considered to be the candidate solutions. Each candidate must define the conditions in which it should be able to contribute to an application's optimization and to what extent these conditions are strictly required, or rather optional. As in the standard LSP approach, this degree of desirability is expressed in terms of a GCD.

The available stacks of progressive enhancement layers are in turn individually evaluated by matching their desired conditions to the mobile device's specific capabilities (e.g., available interaction methods, web technology support, etc.). The incorporation of this LSP step in a mobile context requires a defined set of mobile-relevant mapping functions. The mapping functions specify the similarity between performance variables and the actual device capabilities. To illustrate the concept, both Table I and II contain an implementation of a mapping function that compares the performance variable "stylus interaction" with a device's actual interaction method. The function in Table I uses Boolean logic, which implies that only a perfect match is scored. The one in Table II, on the other hand, uses fuzzy logic. The latter approach makes much better use of the available scoring interval by also grading the less-than-perfect matches. Such examples highlight the importance of carefully thought through mapping functions. In this context, the W3C Mobile Web Best Practices Working Group, as well as Web Accessibility Initiative contributed significantly with their efforts in the mobile web usability and accessibility areas [17][18]. The published set of recommendations is an excellent example of a potential source from which we are able to extract usable mapping functions.

Furthermore, the elementary scores that resulted from matching the candidate progressive enhancement stacks with the device's specific capabilities are aggregated into an overall suitability score. The overall score is attained by applying the candidate's predefined GCD network. Once all

Table I
SAMPLE OF A BOOLEAN LOGIC MOBILE MAPPING FUNCTION

Interaction capability	Match
Touch	0 %
Stylus	100 %
Joystick	0 %
Click wheel	0 %

Table II
SAMPLE OF A FUZZY LOGIC MOBILE MAPPING FUNCTION

Interaction capability	Match
Touch	75 %
Stylus	100 %
Joystick	30 %
Click wheel	10 %

 ISBN: 978-1-61208-156-4

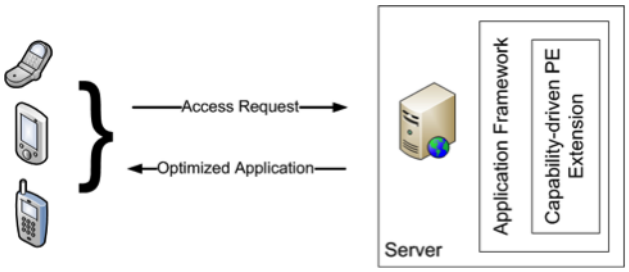


Figure 4. A high level system overview. Integrating the fine-grained progressive enhancement approach within existing web application frameworks

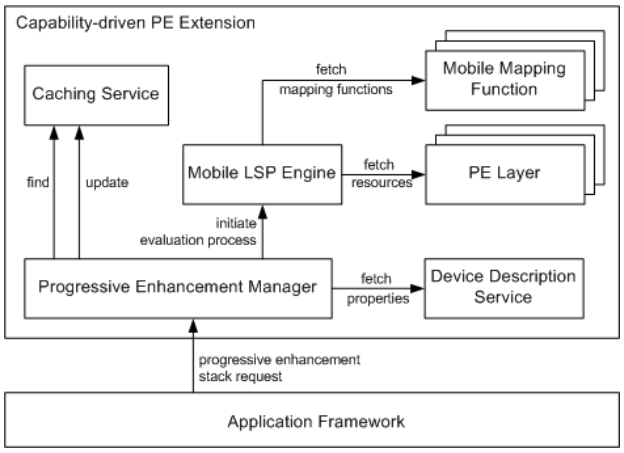


Figure 5. Detailed system architecture. Enabling an application framework to request a stack of progressive enhancement layers that optimally suits the user's device characteristics

combinations of available candidates have been evaluated, the layer selection process is concluded by selecting the highest scoring solution and applying it to the application.

III. Architecture and Design

This section discusses the architectural structure that is needed to integrate the proposed adaptive application composition algorithm within an application development framework. In general, developers dedicate substantial efforts in mastering a specific application framework. It is thus desirable for our system to support existing frameworks rather than introducing a completely new framework. Hence, support for fine-grained mobile progressive enhancement is provided as a generic plug-in for web application platforms. A high level overview of our approach is depicted in Figure 4. The proposed capability-driven progressive enhancement extension interacts with web application frameworks through a web interface. Calling this interface will generate and return a progressive enhancement stack that is optimized for the particular capabilities of the end-user's device. In result, application developers are no longer required to manually support the wide range of mobile variability points, as mobile fragmentation issues are handled by the proposed framework extension. What follows is a short elaboration on the main architectural components of our system, which are shown in Figure 5.

The progressive enhancement manager is the system's interface to the outside world. It delegates all incoming requests from the application framework to compose an optimized stack of progressive enhancement layers. The manager starts by accessing the Device Description Service component, which returns detailed information on the current device context. The contextual description contains data regarding the capabilities and features of mobile devices (e.g., supported CSS and JavaScript versions, available device APIs, etc.). In turn, the manager addresses the Caching Service component to find out whether the result for this particular device has previously been stored. Due to performance considerations, the caching of results for popular devices is an important part of the system (see performance evaluation discussion in Section IV). In case of a cache-miss, the adaptive application composition algorithm will have to be executed. This process is managed by the Mobile LSP Engine component.

The Mobile LSP Engine is responsible for establishing an optimal progressive enhancement stack. This component is at the heart of the proposed system, as it objectively evaluates and combines the applicability of candidate progressive enhancement layers. The engine starts by fetching all available progressive enhancement layers. Next, the mobile mapping functions are retrieved, specifying the degree of similarity between desired device properties and the actual device capabilities. The Mobile LSP Engine calculates the overall similarity score for each candidate progressive enhancement stack. Finally, the engine then selects the stack with the highest overall score, thus the combination of progressive enhancement layers that best matches the capabilities of the client's device. This final selection is passed back to the Progressive Enhancement Manager component, which will cache the result and deliver it to the application framework.

IV. Evaluation

This section evaluates the algorithm and system architecture introduced in Section II and Section III respectively. The proposed capability-driven progressive enhancement extension has been implemented for both the Drupal and Joomla web application frameworks [5][8]. The prototype implementations are used to validate our adaptive application platform objectives and to perform a series of usability and performance evaluations.

A. Proof of Concept Implementation

All architectural components have been implemented for the system's prototype. The repository of device characteristics in the system's Device Description Service component is realized by interfacing with the Wireless Universal Resource File (WURFL) [19]. WURFL is an open source project

 ISBN: 978-1-61208-156-4

gathering information on a vast range of mobile delivery contexts. Furthermore, various mobile progressive enhancement layers were created, as a means to validate the system's capability to adapt to the characteristics of heterogeneous delivery environments. The created enhancements range from simple CSS styling for feature phones, to complex HTML5 and JavaScript layers for high-end smartphones.

B. Usability Evaluation

To evaluate the usability of applications enabled by our system, an adaptive m-commerce application has been built on top of the prototype framework implementations [12]. The application was evaluated using a set of automated usability tests from the W3C MobileOK test suite [16]. The MobileOK service checks the usability of web applications in a mobile context. Tests are based on the validation of markup, accessibility, content and navigation structuring, load time, and the use of network resources. The prototype m-commerce application scores perfect on all MobileOK test. Moreover, the application attains a score in the top 10th percentile of all web applications checked by this W3C service [14].

Applying our fine-grained progressive enhancement approach drastically facilitates the development of accessible and usable mobile applications. Application developer are only required to define of a basic version of their applications, intended for devices with the lowest common denominator (LCD) of capabilities. Figure 6(a) shows the basic LCD structure of the m-commerce application, containing only a simple HTML markup. For more capable devices, such as high-end feature phones, the system detects the applicability of elementary CSS and JavaScript layers. Figure 6(b) depicts this scenario. The LCD version is automatically enhanced with presentational as well as behavioral layers. Furthermore, the system is capable of adapting itself to the capabilities of smartphone devices. The advanced HTML5, CSS3 and AJAX (Asynchronous JavaScript and XML) support allows the system to select complex enhancement layers that mimic an operating system's native look-and-feel (see Figure 7(a) and 7(b)).

C. Performance Evaluation

The adaptive application composition algorithm proposed in Section II has a significant influence on the performance of our approach. As the algorithm evaluates the applicability of all possible progressive enhancement stack combinations, running time increases exponentially with the number of candidate progressive enhancement layers in the system. Moreover, if the system contains a set of n candidate progressive enhancement layers, the algorithm is expected to consume

$$O\left(\sum_{i=0}^{n}\binom{n}{i}\right) = O\left(2^n\right) \tag{3}$$

time for completing the evaluation process. Performance tests on the prototype implementation confirm this prediction. Figure 8 shows the average response times of our platform in function of the number of available progressive enhancement layers. Two types of tests are performed. In the first series of tests, caching is disabled, resulting in rapid performance degradation due to the algorithm's computational complexity. For the second series of tests, on the other hand, the system's caching capabilities are enabled. The use of a caching mechanism drastically improves performance of our system. Results show that in case of a cache hit,

(a) (b)

Figure 6. Adaptive mCommerce web application on two feature phones. (a) The Motorola RAZR, a low-end feature phone and (b) the Nokia N96, a high-end feature phone

(a) (b)

Figure 7. Adaptive mCommerce web application on two smartphones. (a) An Android smartphone, the HTC Dream and (b) an iOS smartphone, the Apple iPhone

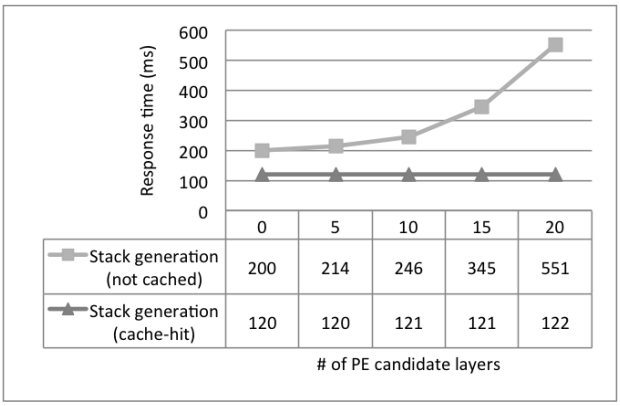


Figure 8. Performance testing results. Response time measured in function of the number of candidate progressive enhancement layers in the system

the generation of a progressive enhancement stack only requires a constant execution time, regardless of the number of candidates that have to be evaluated.

V. Conclusion and Future Work

In this paper, we introduced a platform in support of developing and delivering adaptive mobile web applications. The proposed method can serve as a basis for developers to create and maintain a single version of their mobile application, without being limited by fragmentation related issues. Our adaptive application composition algorithm is at the hearth of the proposed approach. The algorithm is based on a quantitative evaluation algorithm derived from the Logic Scoring of Preference (LSP) method. The proposed approach enables the automated and fine-grained progressive enhancement of web applications. The process is entirely driven by the characteristics of the user's device, in order to provide an optimal user experience.

While the extensive evaluation of our approach has yet to be carried out, initial testing of prototype implementations showed promising results. Future work includes the further validation of our proposed approach as well as the extension of our algorithm towards supporting real time application request handling. Other necessary steps in the development of this application composition method are related to broadening the scope on supported device types. The diversity of devices that enable access to third party software applications is currently extending towards home entertainment systems, automotive, etc. Such evolution further emphasizes the impact of fragmentation on application developers. This is why the applicability of our proposed application composition method should be evaluated for more ubiquitous environments.

References

[1] T. Ahonen. *Mobile As 7th of the mass media: Cellphone, cameraphone, iPhone, smartphone.* London: Futuretext, 2008.

[2] I. Batyrshin, O. Kaynak, and I. Rudas. "Generalized conjunction and disjunction operations for fuzzy control." in *Proc. of 6th European Congress on Intelligent Techniques and Soft Computing EUFIT'98*, 1998, pp. 52-57.

[3] J.J. Dujmovic. "A method for evaluation and selection of complex hardware and software systems." in *Proc. of Int. Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems*, 1996, pp. 368-378.

[4] J.J. Dujmovic, G. De Tre, and N. Van de Weghe. "LSP suitability maps." *J. Soft Computing*, vol. 14, pp. 421-434, 2010.

[5] "Drupal." Available: http://www.drupal.org [Sep. 1, 2011].

[6] G.R. Frederick, and R. Lal. "The future of the mobile web." in *Beginning smartphone web development*, Springer, 2009, pp. 303-313.

[7] International Telecommunication Union. *Measuring the Information Society 2010.* Geneva, Switzerland: ITU, 2010.

[8] "Joomla." Available: http://www.joomla.org [Sep. 1, 2011].

[9] G. Kappel, B. Proll, W. Retschitzegger, and W. Schwinger. "Modeling ubiquitous web applications: the WUML approach." in *Conceptual Modeling for New Information Systems Technologies*, Springer, 2002, pp. 183-197.

[10] N. Koch, A. Knapp, G. Zhang, and H. Baumeister. "UML-Based web engineering." in *Web engineering: modeling and implementing web applications*, Springer, 2008, pp. 157-191.

[11] A. Schauerhuber, M. Wimmer, M. Schwinger, E. Kapsammer, and W. Retschitzegger. "Aspect-oriented modeling of ubiquitous web applications: the aspectWebML approach." in *Proc. of Int. Conference on the Engineering of Computer-Based Systems*, 2007, pp. 569-576.

[12] "WAFL Adaptive e-commerce demonstration." Available: http://www.wafl.ugent.be/demonstration [Sep. 1, 2011].

[13] J. Wells, and C. Draganova. "Progressive enhancement in the real World." in *Proc. of 18th conference on Hypertext and hypermedia HT2007*, 2007, pp. 55-56.

[14] W3C. "MobileOK checker usage statistics." Available: http://www.w3.org/2008/04/mokstats/public/global/20080401-20100110/Overview.html [Sep. 1, 2011].

[15] H.W. Lie, T. Celik, D. Glazman, and A. van Kesteren. "Media Queries." Available: http://www.w3.org/TR/css3-mediaqueries [Sep. 1, 2011].

[16] J. Rabin, and P. Archer. "MobileOK Scheme." Available: http://www.w3.org/TR/mobileOK [Sep. 1, 2011].

[17] W3C. "Mobile Web Best Practices Working Group." Available: http://www.w3.org/2005/MWI/BPWG [Sep. 1, 2011].

[18] W. Chrisholm, G. Vanderheiden, and I. Jacobs. "Web Content Accessibility Guidelines." Available: http://www.w3.org/TR/WAI-WEBCONTENT [Sep. 1, 2011].

[19] L. Passini. "Wireless Universal Resource File." Available: http://wurfl.sourceforge.net [Sep. 1, 2011].

 ISBN: 978-1-61208-156-4

LTE Uplink Power Control and its Impact on System Performance

Elena-Roxana Cîrstea, Silviu Ciochină
Electrical Engineering and Telecommunications
'Politehnica' University of Bucharest
Bucharest, Romania
rcirstea@elcom.pub.ro, silviu@comm.pub.ro

***Abstract*—Long Term Evolution (LTE) is the new standard of the 3GPP (3rd Generation Partnership Project) and it was designed to increase capacity and improve service performance. Since SC-FDMA (Single Carrier-Frequency Division Multiple Access) is the multiple access scheme being used, compared with WCDMA (Wideband Code Division Multiple Access) multiple access scheme implemented in UMTS or HSPA, there are no intra-cell interferences. The problem of inter-cell interference is still present. One of the fundamental algorithms that was developed to control inter-cell interferences is the uplink fractional power control mechanism. This paper evaluates possible ways to tune this adaptive algorithm in order to maximize system performance. Simulation results are presented in order to identify the best option for algorithm parameter setting.**

Keywords - power control; adaptive algorithm; throughput; load; SINR (Signal to Interference Plus Noise Ratio)

I. INTRODUCTION

Over the last few years, the requirements concerning service performance have become very tight. To be able to deliver the strict performance requirements requested by applications and services, new standards for innovative technologies are under development. For many operators in many countries, this year is going to be the starting point for launching LTE (Long Term Evolution) networks. The main objectives for this new system are: improvement of spectral efficiency, increase of uplink and downlink transfer rates, support for scalable bandwidth and all IP network, reduced delay.

The uplink multi access scheme is SC-FDMA (Single Carrier FDMA). By using this access scheme, a unique frequency can be allocated to users at the same time. This way, intra-cell interference is eliminated. Also, carriers are not combined in random phases so there is no large variation of the instantaneous power for the modulated signal. This translates into a lower PAPR (Peak to Average Power Ratio). The power amplifier in the terminal has an increased efficiency and power consumption is reduced.

One of the key elements that is in charge to improve service performance is the Uplink Fractional Power Control (FPC) [1]. As it is known, interference, even if it is intra-cell or inter-cell, it is an important factor that introduces severe service performance degradation and users dissatisfaction. The FPC algorithm was designed to improve cell throughput or cell edge throughput and to improve battery life time [2],[3].

This paper investigates the way FPC mechanism works and evaluates possible tuning of this algorithm and its impact on service performance both on user experience and on system functionality.

The paper is structured in four parts. The first part is dedicated to a short introduction and the second part describes the fractional power control algorithm as it is specified in the standard. The last two parts are the main parts of the paper that contain simulated scenarios and relevant results. Finally, after analyzing simulation results, recommendations concerning parameter setting are provided.

II. FRACTIONAL POWER CONTROL ALGORITHM

In LTE system, the power control algorithm for uplink is different, compared to that used for UMTS and HSPA. The differences come from specific characteristics of the LTE system like:

- The multiple access scheme for LTE uplink is SC-FDMA (WCDMA for UMTS) so there are no intra-cell interferences thanks to user orthogonality assured by the multiplexing scheme;
- The LTE flat architecture where base stations communicate one to another to control inter-cell interferences (RNC –Radio Network Controler- the superior entity in UMTS and HSPA that controls interactions between base stations is eliminated in LTE and eNodeB (enhanced NodeB) takes part of its responsibilities);
- The uplink scheduler is located at basestation level and power control should synchronize with link adaptation to be able to adapt to multiple QoS requirements of applications.

Uplink power control mechanism is part of link adaptation mechanisms [5]. Its purpose is to compensate channel variations. Based on different channel variations, two power control categories have been defined:

a) Slow Power Control: compensates for slow channel variations (pathloss, shadow fading) *;*

b) Fast power control: compensates for fast channel variations, like fast fading;

Another classification for power control algorithms is based on the characteristics of the information sent by the mobile terminal to establish its level of transmit power:

a) Open Loop Power Control: power level is determined by the terminal using parameters and mesurements obtained from the signals sent by eNodeB; no feedback related to the power being used by the user equipment is sent to the eNodeB;

 ISBN: 978-1-61208-156-4

b) Closed Loop Power Control: the mobile terminal sends its feedback to the eNodeB which is used afterwards for power corrections.

A Closed Loop Power Control scheme is going to introduce important overhead but at the same time, it will be able to do a better compensation for channel variations. Open Loop Power Control schemes are easy to implement and signalization overhead is much lower compared to the anterior scheme. The limitation is that power compensation for channel variations cannot be done at terminal level.

Full compensation power control schemes implemented for UMTS and HSPA uplink assume that all users are received with the same SINR. In a real network, users have different SINR requirements. For example, users located at cell edge have a higher path loss and instead of using full compensation which increases interference of the neighboring cells, fractional power control scheme is being used. This way, cell edge users could operate at lower SINR and interferences at neighboring cells are diminished.

The standardized formula for uplink fractional power control [1] is described in the next equation:

$$P = min\{P_{Max}, P_0 + 10log_{10}M + \alpha PLt + \delta_{mcs} + f(\Delta_i)\} \quad (1)$$

P_{MAX} is the maximum transmit power of the terminal and it depends on terminal power class, M is the number of PRBs (Physical Resources Blocks) being allocated by the uplink scheduler on PUSCH (Physical Uplink Shared Channel), P_0 is a specific cell parameter that represents the power allocated to one PRB, α is also a cell parameter, PL is the downlink pathloss measured by the mobile terminal, δ_{MCS} is an offset specific to an user equipment and it is in a strict connection with the modulation and coding scheme and the last parameter, $f(\Delta_i)$ is a function that allows relative, cumulative and absolute corrections and it is also user specific.

The eNodeB broadcasts parameters P_0 and α. The destination of this broadcast is represented by the user equipment. Using these two parameters plus the measured pathloss, a mobile terminal is able to establish the power level used during the first phase. This is the Open Loop Power Control. After this stage, mobile terminals send feedback to the eNodeB. δ_{MCS} and Δ_i are being sent by the eNodeB. These last two terms represent the Closed Loop Power Control.

When α=0, no power control mechanism is implemented and all mobile terminals use the same transmit power. The case when α=1, corresponds to full power compensation. For α in the interval 0 and 1, a compromise is being done in between full compensation and no power control mechanism. In this case, a fractional compensation of pathloss is used.

As it was specified, parameter α can take values in the set {0, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1} although values in between 0 and 0.4 have no practical use.

While the value for parameter α determines a compromise between cell edge users and cell center users, P_0 equally impacts all the users in the cell. As α decreases, cell edge throughput decreases and cell average throughput increases [2], [3], [4]. Each possible combination of the parameters α and P_0 defines an operation point for the Fractional Power Control algorithm and also a certain performance.

For this study, Closed Loop Power Control is not taken into account and only Open Loop Power Control is analyzed. This mechanism is based on the principle that target SINR is calculated using pathloss. Actually, SINR is a linear function of pathloss. Therefore, it is possible to have a higher SINR target at cell center in order to increase throughput while having a lower SINR target at cell edge. The low SINR target at cell edge it is an important factor destined to decrease the level of inter-cell interference.

The Fractional Power Control algorithm allows a tradeoff between cell center users and cell edge users. When α is less than 1, the SINR target decreases with pathloss. As pathloss decreases, a smooth increase in SINR target was observed.

The combination of Open Loop and Closed Loop Power control is in charge to correct the open loop errors.

III. SIMULATED SCENARIOS

To be able to evaluate and provide recommendation in such a way that system performance is maximized, several scenarios have been run.

For each run, the mobiles are placed on the geographical area and rejected or admitted according to specified radio conditions. After this stage, the performance per mobile station is calculated. At the beginning of a new simulation, there are mobiles present in the system and also new mobiles arrive in the network. After identifying the best server for each mobile terminal, the admission control is performed.

Here are the inputs for the system-level simulator:

- Frequency band: 2.6GHz
- Bandwidth: 10MHz
- Environment: Suburban
- Traffic: FTP (File Transfer Protocol)
- Slow power control: compensate for slow channel variations (pathloss, shadow fading)
- Path Loss model: Hata suburban, eNodeB antenna height 30m, mobile terminal height 1.5m
- Shadow fading: σ=6dB, δ=0.95
- Maximum transmit power for the terminal: 23dBm
- Fading channel profile: 3km/h, Extended Pedestrian
- The nominal pathloss related to the reference target SINR: 60dB
- No mobility

The number of Physical Resource Blocks (PRB) allocated to a user depends on the service type. For FTP traffic the user requests a minimum guaranteed rate and it has no delay constraints. The number of PRBs allocated is determined by the value of the minimum guaranteed rate.

The uplink power control mechanism is adjusted for each modulation scheme and coding rate. Power adjustment is done while maintaining a target BLER. The first step is to calculate the SINR achieved with maximum transmit power, then the modulation scheme corresponding to the predetermined SINR is decided. The last step is to calculate

 ISBN: 978-1-61208-156-4

the lowest power that can be used to obtained the modulation and the coding scheme previously determined.

Admission control is based on radio conditions and the number of available PRBs. A mobile could be rejected based on uplink or downlink coverage or the unavailability of free PRBs. For uplink admission, each mobile has to reach a C/I threshold to meet the QoS requirements. When the required power to meet this threshold is higher than the maximum transmit power of the terminal, the user is rejected due to bad uplink coverage. Then, the number of PRBs is verified. If the number of unallocated PRBs is lower that the required number to offer the required QoS, the user is blocked and it is taken into account for blocking statistics.

For each simulation, the parameters related to the Fractional Power Control algorithm that can be modified are:

- Parameter α
- Target SINR value at cell edge, Min_SINR
- Target SINR value at cell center, Max_SINR
- Arrival rate of users in the system

In order to evaluate the impact of the Fractional power control parameter setting on system performance, the following scenarios have been considered for the suburban case. The table below contains the proposed scenarios:

TABLE I. SIMULATED SCENARIOS

Scenario number	α	Min_SINR	Max_SINR	Arrival rate	Traffic
1	0.8	-1dB	12dB	80	UL+DL
2	0.5	-1dB	12dB	80	UL+DL
3	0.8	2dB	12dB	80	UL+DL
4	0	-1dB	12dB	80	UL+DL
5	0.8	2dB	12dB	200	UL+DL
6	0	-1dB	12dB	200	UL+DL
7	0.8	5dB	12dB	80	UL+DL
8	0.7	-1dB	12dB	80	UL+DL
9	0.8	-5dB	12dB	80	UL+DL
10	0	-1dB	12dB	120	UL+DL
11	0.7	0dB	12dB	80	UL+DL
12	0.8	2dB	12dB	120	UL+DL
13	0.7	-1dB	12dB	120	UL+DL
14	0.8	5dB	12dB	120	UL+DL
15	0.7	-5dB	12dB	80	UL+DL
16	0	8dB	8dB	80	UL
17	0	-1dB	12dB	120	UL
18	0.8	2dB	12dB	80	UL
19	0.7	0dB	12dB	80	UL

The case when α is 0 means that the Fractional Power Control mechanism is deactivated. Because of the low admission probability due to downlink limitations (not enough resources to accommodate all downlink users), downlink traffic is set to 0 for the last four scenarios.

To study the functionality of the Fractional Power Control algorithm, a number of four parameters in the simulation output will be analyzed and compared between all defined scenarios:

a) Admission probability: The probability for a user to be admitted in the network. It is defined as the percentage of admitted users from the total number of generated users.

b) Average uplink load: The ratio between the average number of PRBs allocated for uplink traffic and the number of total available PRBs at eNodeB level.

c) Average uplink throughput at cell level, R:

$$R = \frac{N_{PRB}}{\text{Simulation time}} N_{PRB}^{allocated}(1 - \text{BLER}) \quad (2)$$

where N_{PRB} is the total number of available PRBs on uplink, at eNodeB level, $N_{PRB}^{allocated}{}_{i}$ is the number of PRBs allocated to user i during one simulation run and BLER is the Block Error rate, also an input for the simulation.

d) Average number of users in the cell: Average number of FTP users during a simulation run.

IV. SIMULATION RESULTS

The next stage after deciding the scenarios to be run, is to collect and interpret the obtained results. Figure 1, illustrated on the next page, contains the four parameters after running the proposed scenarios. Comparing different scenarios, the functionality of the fractional power control is analyzed.

Because mobility is not considered, pathloss is constant for each user during one run.

Varying only Min_SINR (scenarios 1, 3, 7, 9, 18) and for α=0.8, when Min_SINR increases, both, the admission probability and the number of user in the cell also increases, uplink load decreases (higher modulation and coding scheme, better spectral efficiency) and the throughput is higher. Comparing scenario 3 and scenario 18 with uplink traffic only (avoid being limited by downlink), the output parameters maintain the same tendency.

For a decreasing α (scenario 1, scenario 8, scenario 2, scenario 4), pathloss compensation is lower so the mobile transmits with less power and SINR decreases. In this case, cell edge throughput decreases and average throughput increases. Also admission probability is lower. To make a compromise between cell edge throughput and average cell throughput, the possible values for α are 0.7 and 0.8, depending on operator deployment strategy.

By analyzing simulation number 16 and comparing it with scenario 4, users are closer to cell center and throughput, admission probability, average number of users are all higher.

Increasing arrival rate while maintaining other parameters at constant values (scenarios 3, 5, 12), admission probability decreases and uplink load increases. This is the case when the Fractional Power Control algorithm is activated. For the case when this algorithm is turned off (scenarios 4, 6, 10), although admission probability and average number of users in the cell is comparable with the case when power control is turned on (simulation output in figure 1 for scenarios 3 and 4, 5 and 6, 10 and 12), uplink load is lower in this case.

When increasing arrival rate, the admission probability has a very small increase for FPC activated (the difference between admission probabilities is increasing by 0.2 for scenario 3 compared to scenario 4, 0.4 for scenario 10 and S12, 0.5 for scenario 5 compared to scenario 6).

 ISBN: 978-1-61208-156-4

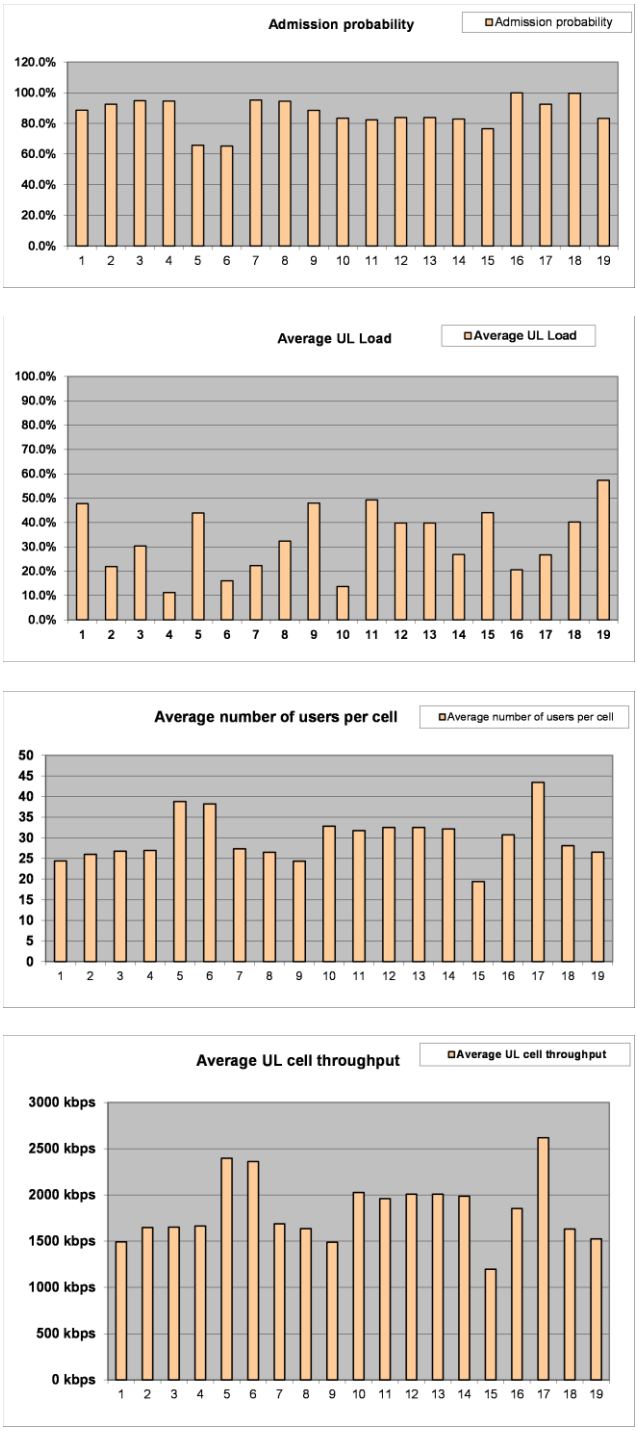


Figure 1. Simulation results: Admission probability, Average uplink load, Average uplink throughput at cell level; Average number of users in the cell

Comparing scenario 3 and scenario 8 (α is 0.8 and 0.7) and arrival rate 80 and scenario12 and scenario 13 (same settings for α) with arrival rate 120, the average throughput per Cell, uplink load and admission probability are higher for the case when alpha is 0.8 and arrival rate is 80. Increasing arrival rate, almost the same results are obtained for both alpha 0.7 and 0.8, the difference in between the two α values translates into a higher cell average throughput.

Another interesting comparison to be done is scenario 9 (α=0.8) and scenario 15 (α=0.7). Both scenarios have the same Min_SINR, -5dB. Although three of the output parameters are comparable (load is only 4% higher for scenario 9), the average throughput for scenario 9 is 1489kbps while for scenario 15 it is only 1196kbps. Analogous, an analysis for scenario 1 (α=0.8) and scenario 8 (α=0.7) but this time Min_SINR is -1dB. The average throughput for simulation 1 is 1493kbps and 1635kbps for simulation 8. Average throughput for the first comparison is approximately 300kbps higher for α=0.7 and only 142kbps higher also for α=0.7 but for higher Min_SINR.

A parallel between scenario 4 (no FPC) and scenario 8 (α=0.7) both with Min_SINR=-1dB shows that throughput is comparable. Because users start to be limited by power, the effect of the Fractional Power Control mechanism is less evident. Conversely, evaluating output from scenario 14 (FPC on, arrival rate 120 and Min_SINR 5dB) and scenario 16 (FPC off, arrival rate 80 and Min_SINR 8dB) throughput is higher when FPC is active. This is the case when user located close to cell center can benefit more from a power increase compared with cell edge users.

Taking into consideration the implemented simulations, the recommendation for parameter α is 0.7. Using this value, it gives an opportunity for the network operator to make a compromise between full pathloss compensation where fairness is taken into account and fractional pathloss compensation where users close to cell center can better benefit from good radio conditions. At the same time, an equilibrium between cell edge and cell center average throughput is maintained.

To better highlight the advantages of the Fractional Power Control, a future study for dense urban should be done.

CONCLUSION

This paper has assessed the functionality of the adaptive fractional power control algorithm for uplink LTE. First, some relevant scenarios have been run and the output parameters have been analyzed.

It can be stated that the uplink power control mechanism has a key role in optimizing uplink system capacity. Uplink power control together with uplink scheduling strategies can enhance cell edge performance by realizing interference coordination. Another way to increase uplink system performance is to implement the Closed Loop Power Control adaptive mechanism to correct errors generated by the Open Loop Power Control algorithm.

ACKNOWLEDGMENT

Roxana Cirstea is a POSDRU grant beneficiary offered through POSDRU/6/1.5/S19 contract.

 ISBN: 978-1-61208-156-4

REFERENCES

[1] 3GPP TS 36.213 v10.1.0, E-UTRA Physical layer procedures, pp. 9–18, March 2011

[2] C. U. Castellanos, D. L. Villa, C. Rosa, K. Pedersen, F. D. Calabrese, P. Michaelsen and J. Michel, Nokia Simens, Performance of Uplink Fractional Power control, in press, May 2011

[3] N. J. Quintero, Aalborg University, Advanced Power Control for UTRAN LTE Uplink, June, 2008

[4] H. Holma and A. Toskala, LTE for UMTS - OFDMA and SC-FDMA Based Radio Access, John Wiley & Sons, 2009, pp. 202-232

[5] S.Sesia, I. Toufik and M. Baker, LTE – The UMTS Long Term Evolution, From Theory to Practice, John Wiley and Sons, pp. 484-492, 2009

[6] K. Parsa, Survey of power control in LTE, http://www.parsawireless.com/casestudies/lte_pc_report_1q_2008.pdf, 2008

 ISBN: 978-1-61208-156-4

Temporal Mechanisms for Communications in Real-Time Networks

Pascal LORENZ
University of Haute Alsace
34, rue du Grillenbreit 68008 Colmar - France
e-mail: lorenz@ieee.org

***Abstract*—Some processes should respect real-time constraints to be valid. Temporal mechanisms should indicate whether time constraints are met or not, and when they are not, these mechanisms should identify the causes of their non-compliance. We present some mechanisms enabling to indicate whether the time constraints were met in real-time and time-critical systems.**

Keywords-Temporal constraints, Time Windows, Real-Time, Modelization, Communication.

I. INTRODUCTION

In real-time environments, the processes consistencies depend of the obtained results but also of the respect of temporal constraints. This imposes that the start and/or the end of execution respect the timing constraints. Systems which must meet strict time constraints to prevent disasters are called time-critical systems [1, 4, 5].

Thus, for example, we should be able to know, why in an industrial plant, when exceeding a critical threshold temperature detected by a sensor at the time t1, an actuator has not triggered the closure of a valve at time t1 + □t as it was intended in the specification. Is this due to the fact that the sensor has not transmitted information within the time limit or this is due to network congestion or this is due to actuator that has not taken into account the information received on time ?

This paper is structured as follows: in section II, we describe the specification of time-windows constraints. Section III gives a detailed description of validity of a variable in a time-window. Section IV presents a modelization of the temporal mechanisms. Conclusion and future work are given in section V.

II. SPECIFICATION OF TIME-WINDOWS CONSTRAINTS

Communication delays in the networks are generally not deterministic, and therefore the data arriving at a consumer may be outdated, since a time-critical data have a lifetime (or validity) determined by the application [6, 7]. The data lifetime for the producer (respectively consumer) is the time interval during which the value produced (respectively consumed) remains valid and is valid for the application. Thus, data generated are valid and make sense for a certain period of time after their production.

We have chosen to represent the time involved during the communication steps in terms of time constraints. We have introduced mechanisms to know if during the communication, some data has been produced, transmitted and consumed in a timely and otherwise, they allow locating the causes of non-compliance with time constraints.

Different types of operators allow taking into account time constraints. Time constraints at the earliest date and latest date can be represented in terms of time intervals. Time window mechanisms take into account time constraints at the earliest date and latest date, and time constraints are expressed as the maximum time for a communication between entities [3].

The various possible combinations between two times intervals are described in the diagram below:

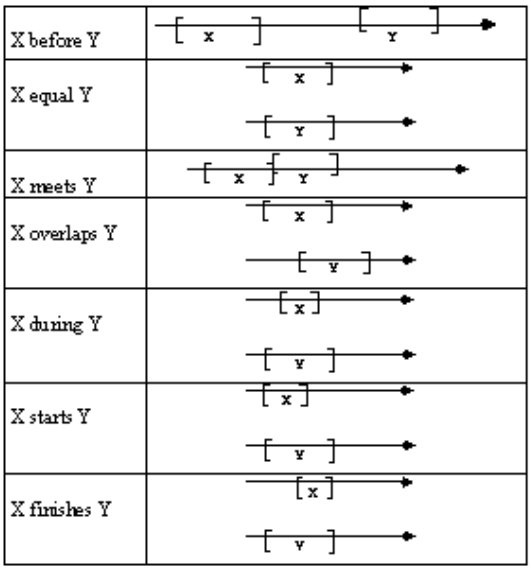


Fig. 1: Relationships between time intervals

We always assume that local clocks entities of producing and consuming entities are synchronized and that the time

 ISBN: 978-1-61208-156-4

difference between all the entities producing and consuming is negligible compared to the nature of the application. Thus, beginnings and ends of time windows will be fixed by using physical clocks synchronized [2, 8, 9].

A time window, denoted TW thereafter, has three parameters:
- start time,
- ending time and
- duration.

These three parameters can be preset or not.

↑ represents the start time, ↓ the end and Δ the length (or duration) of the time window. Let x be a time window, if all parameters are known, then one can verify that ↑x + Δx = ↓x.

We will use the parameters i and v to represent the i^{th} time window of activity used for the transmission of a variable v.

For each new production, transmission, reception or consumption of a given variable v, a new time window is used by incrementing the index i of the TW.

A time window can be described as follows:

x y (v, i) with x = (↑, ↓ or Δ) and
y = (prod, send, rec or cons).

We will now present a set of rules that must always be taken into account to allow a consistent sequencing of the various TW.

The fact that the start of the TW is always prior to the end of the same TW (production, transmission, reception or consumption) can be represented by the following rule R1:

R1: ↑y (v, i) < ↓y (v, i)

If we apply R1 to the different transmission sequences, we obtain the following rules R2 and R3:

R2: ↑prod (v, i) < ↑send (v, i) < ↑rec (v, i) < ↑cons (v, i)

R3: ↓prod (v, i) < ↓send (v, i) < ↓rec (v, i) < ↓cons (v, i)

If we have only earliest constraints date, there is no need to chain the various TW, since it is only enough to start on time.

The sequencing of the various TW can be done by using either fixed or slippery TW.

Fixed TW allows managing earlier and latest constraints dates. On the other side, slippery TW can only manage latest constraints dates.

These can be defined as follows:

- Fixed TW: early and latest time intervals are fixed once and for all. Thus, for a production entity, if a production is done more or less early in the time slot allocated, it does not affect the start date for sending the variable. Similarly, in the consumer entity, the fact that a variable is received earlier or later has no effect for the start of the consumption of this received variable.

- Slippery TW: there is no time constraint on the start of different time intervals involved in the communication. Thus, as soon as data is produced within the constraints of time, it offers the possibility of the sending this later. Similarly, as soon as a data is received within the time constraints, the consumer can consume this later.

Fixed TW can be described as follows:

[↑prod (v, i), ↓prod (v, i)] before [↑send (v, i), ↓send (v, i)]
[↑send (v, i), ↓send (v, i)] overlaps [↑rec (v, i), ↓ rec (v, i)]
[↑rec (v, i), ↓rec (v, i)] before [↑cons (v, i), ↓cons (v, i)]

The slippery TW can be described as follows:

[↑prod (v, i), ↓prod (v, i)] overlaps [↑send (v, i), ↓send (v, i)]
[↑send (v, i), ↓send (v, i)] overlaps [↑rec (v, i), ↓rec (v, i)]
[↑rec (v, i), ↓rec (v, i)] overlaps [↑cons (v, i), ↓cons (v, i)]

Thus for fixed TW, the following rules R4 and R5 indicate the links between the communication steps:

R4: ↓prod (v, i) <↑send (v, i)
R5: ↓rec (v, i) <↑cons (v, i)

The different cases of sequence for fixed TW and sliding TW can be described as follows:

- Sequence of sliding windows = R1 + R2 + R3
- Sequence of fixed windows = R1 + R2 + R3 + R4 + R5
- Sequence of fixed windows at the producer and the sliding window at the consumer = R1 + R2 + R3 + R4
- A series of sliding windows at the producer and fixed windows at the consumer = R1 + R2 + R3 + R5

↑y (v, i +1) must be always higher than the beginning of the previous TW, represented by the rule following R6:

R6: ↑y (v, i +1) > ↑y (v, i)

 ISBN: 978-1-61208-156-4

III. TIME-WINDOW FOR THE VALIDITY OF A VARIABLE

A variable produced at a given moment has duration of life that varies depending on the consumers. Thus in the case of an industrial application, for an application process that makes statistics, some data can be considered valid with an infinite duration. On the other side, for an application process associated with an actuator operating in a real time environment, this same data will be, for example, valid for only 20 ms. Therefore, the notion of data lifetime depends on the use made of it.

To represent the data's lifetime, we will again use the time window mechanism. We will now try to formalize the concept of lifetime of a variable in the same way as we did for the operations of production, transmission, reception and consumption. A time window for the lifetime (or the validity) of a variable will be denoted VW.

We will use the same notation $\uparrow$, $\downarrow$ and Δ used by the TW. The parameters v and i will be used to represent, respectively, the i^{th} VW instantiated for the production of a variable v in a TW.

VW is always associated with a given TW, since the beginning of a VW takes always place in a production TW. Moreover, it is important to note that it is possible to have multiple productions of a given variable even within a single production TW.

When a variable is produced, it has a lifetime, denoted Δ, and it has a lifetime Δ' (with $\Delta'<\Delta$) when it is received by the consumers.

A time window on the validity of a variable is described as follows:

x val (v, i) with x = ($\uparrow$, $\downarrow$ or Δ)

$\uparrow$val (v, i) represents the moment when the variable v is produced and $\downarrow$val (v, i) represents the end of the validity of the variable v.

The various parameters of VW should generally be defined by the user. As for the TW, the beginning of a VW is always prior to the end of this VW. This translates into the following rule R7:

R7: $\uparrow$val (v, i) < $\downarrow$val (v, i)

As the production of a variable is always at the beginning of VW, it must necessarily be in the production TW since the purpose of a production TW is to verify that the data has been produced on time. This case is represented as follows:

R8: $\uparrow$prod (v, i) <$\uparrow$ val(v, i) <$\downarrow$prod (v, i)

For the end of the VW, the two main cases can appear:

- the end of the VW appears before the end of the TW consumption
- the end of the VW appears after the end of the TW consumption.

When the end of the VW occurs before the end of the consumption TW is represented as follows:

[$\uparrow$val (v, i), $\downarrow$val (v, i)] finishes [$\uparrow$prod (v, i), $\downarrow$cons (v, i)]

which is a special case of:

[$\uparrow$val (v, i), $\downarrow$val (v, i)] during [$\uparrow$prod (v, i), $\downarrow$cons (v, i)]

When the end of the VW occurs after the end of the consumption TW is represented as follows:

[$\uparrow$prod (v, i), $\downarrow$cons (v, i)] overlaps [$\uparrow$val (v, i), $\downarrow$val (v, i)]

Depending on the variable's lifetime, it is possible to transmit this information sooner or later, while issuing a valid data to the consumer entity.

Similarly in the consumer entity, depending on the variable's lifetime, it is possible to consume this data sooner or later while using valid data.

As the lifetime of a variable should allow to the production and consumption entities to know at any time if the value of a given variable is valid or not, it is necessary to introduce two cases:

- in the first case, the lifetime of a variable is known only locally by each consumption entities. Then only $\downarrow$val (v, i) enable to know when a certain variable is no longer valid.

- in a second case, the lifetime of a variable is set globally once and for all, then the consumption entities should calculate the remaining validity when it receives the variable.

We realize the importance of the clocks synchronization. If the local clocks of the producer and the consumer are not synchronized, then the notion of validity cannot be taken into account by the producing and consuming entities.

In the case where a single variable is used by several consuming entities, if the different local clocks are not synchronized, then the validity of this variable will expire at different times in different consuming entities.

IV. MODELISATION OF TEMPORAL MECHANISMS

At a given time, it is important to know if the current state is located inside or outside a TW and a VW; and to know if VW (v, i-1) exists or not.

The different possible situations for a TW and a VW are:

 ISBN: 978-1-61208-156-4

- located out of a TW, out of a VW and it does not exist a previous VW for a given variable v,
- located in a TW, out of a VW and it does not exist a previous VW for a variable v,
- located in a TW and in a VW; this state is called True state,
- located in a TW, out of a VW and it exists a previous VW for a variable v,
- located out of a TW and in a VW
- located out of a TW, out of a VW and it exists a previous VW for a variable v.

Table 1 describes the different possibilities with the previous proposed notation:

In TW (v, i)	In VW (v, i)	∃ a VW (v, i-1)	Name of the state
0	0	0	False
0	0	1	Expiration2
0	1	-	TW-Expiration1
1	0	0	Wait
1	0	1	VW-Expiration1
1	1	-	True

Table 1. Description of the different states

The symbols used in table 1 are:

0 = the condition is not verified
1 = the condition is verified.

Then the six possible combinations between the TW and VW are:

- 1 = False state: located out of a [Start Function (v, i), End Function (v, i)], out of a [Start VW (v, i), End VW (v, i)] and it does not exist a Start VW (v, i-1)

- 2 = Wait state: inside the a [Start Function (v, i), End Function (v, i)], out of the [Start VW (v, i), End VW (v, i)] and it does not exist a Start VW (v, i-1)

- 3 = True state: inside the [Start Function (v, i), End Function (v, i)], inside the [Start VW (v, i), End VW (v, i)], then at least one variable v has been produced respecting the timing constraints

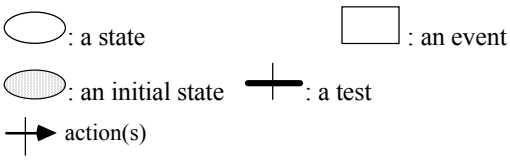
- 4 = VW-Expiration1 state: inside a [Start Function (v, i), End Function (v, i)], out of a [Start VW (v, i), End VW (v, i)] and it exists a Start VW (v, i-1)

- 5 = TW-Expiration1 state: out of a [Start Function (v, i), End Function (v, i)], but inside [Start VW (v, i), End VW (v, i)]

- 6 = TW+VW-Expiration / Expiration2 state: out of the [Start Function (v, i), End Function (v, i)], out of a [Start VW (v, i), End VW (v, i)] and it exists a Start VW (v, i-1).

For each communication step, we will have a start event (generating an event to start the TW and the VW) and an end event (generating a time-out to close the TW and the VW).

The protocol integrating the 6 different states of the TW and VW is described in the following figure 2. The symbols used in figure 2 are:



When the application start, the initial state is the False state.

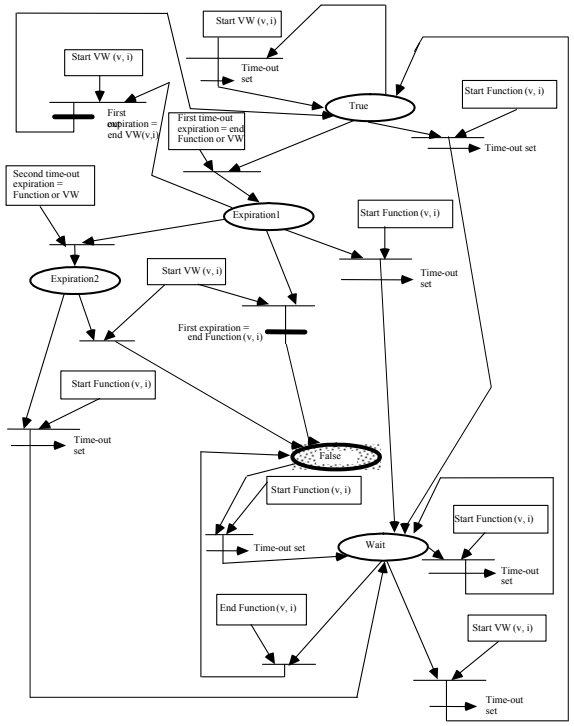


Fig. 2: Description of the protocol integrating Temporal Window and Variable validity time Window

V. CONCLUSION

We studied how time could be represented for real-time communication. We have introduced the concepts of time windows for production, transmission, reception and consumption, as well as concepts of time windows for the validity of variables in order to formalize the concepts of variables' lifetime.

We have proposed a set of rules to bind the different time windows and the time window for variables' validity.

In future work we will develop a prototype which supports the temporal mechanisms for real-time communication processes.

 ISBN: 978-1-61208-156-4

REFERENCES

[1] Brooke Shrader, Armen Babikyan, Nathaniel M. Jones, Thomas H. Shake, Andrew P. Worthen, "Rate Control for Network-Coded Multipath Relaying with Time-Varying Connectivity", IEEE Journal on Selected Areas in Communications, Vol. 29, no. 5, May 2011

[2] María José Pérez-Luque, Thomas D. C. Little, "A Temporal Reference Framework for Multimedia Synchronization", IEEE Journal on Selected Areas in Communications, Vol. 14, no. 1, January 1996, pp. 36-51

[3] Richard L. Schwartz, P. Michael Melliar-Smith, "From State Machines to Temporal Logic: Specification Methods for Protocol Standards", IEEE Transactions on Communications, Vol. 30, no. 12, December 1982, pp. 2486-2496

[4] Vijay Arya, N. G. Duffield, Darryl Veitch, "Temporal delay tomography", IEEE INFOCOM 2008 - 27th IEEE International Conference on Computer Communications, Vol. 27, no. 1, April 2008, pp. 495-503

[5] Chih-Jen Tseng, Chyou-Hwa Chen, "Exploiting the temporal dimension in QoS-aware multicast routing", ICC 2002 - IEEE International Conference on Communications, no. 1, April 2002, pp. 1284-1289

[6] Juan Jose Jaramillo, R. Srikant, Lei Ying, "Scheduling for Optimal Rate Allocation in Ad Hoc Networks With Heterogeneous Delay Constraints", IEEE Journal on Selected Areas in Communications, Vol. 29, no. 5, May 2011

[7] Ye Ge, Jennifer C. Hou, Hung-Ying Tyan, "A packet eligible time calculation mechanism for providing temporal QoS for multicast routing", ICC 1999 - IEEE International Conference on Communications, no. 1, June 1999, pp. 721-726

[8] Ahmet Feyzi Ates, Murat Bilgic, Senro Saito, Behcet Sarikaya, "Using Timed CSP for Specification Verification and Simulation of Multimedia Synchronization", IEEE Journal on Selected Areas in Communications, Vol. 14, no. 1, January 1996, pp. 126-137

[9] Ian F. Akyildiz, Wei Yen, "Multimedia Group Synchronization Protocols for Integrated Services Networks", IEEE Journal on Selected Areas in Communications, Vol. 14, no. 1, January 1996, pp. 162-173

 ISBN: 978-1-61208-156-4

A Formal Orchestration Model for Dynamically Adaptable Services with COWS

Jorge Fox
Department of Telematics
Norwegian University of Science and Technology
Trondheim, Norway
jfox@item.ntnu.no

***Abstract*—The growing complexity of software systems, as well as, changing conditions in their operating environment demand systems that are more flexible, adaptable and dependable. In many domains, adaptations may occur dynamically and in real time. In addition, services from heterogeneous, possibly unknown sources may be used. This motivates a need to ensure the correct behaviour of the adapted systems, and its continuing compliance to time bounds and other Quality of Service properties. The complexity of Dynamic Adaptation is significant, but currently not well understood or formally specified. This paper elaborates a well-founded model of dynamic adaptation, introducing formalisms written using the process algebra COWS. The model provides the foundation for exploring dynamic adaptation and assessing it against predefined specifications. We consider it a contribution for the design of new models and methodologies for system adaptability.**

***Keywords*-Dynamic Adaptation; Formal Languages; Formal Methods.**

I. INTRODUCTION

Modern software systems typically operate in dynamic environments and are required to deal with changing operational conditions, while remaining compliant with the contracted level of service. The execution context of modern distributed systems environments is not static but fluctuates dynamically, and to provide the expected functional service with the desired qualities, systems need to be adaptable. Software service adaptation supports modification of existing services or inclusion of new ones, in response to inputs from the operating environment. Inputs or triggers for adaptation include changes in the running environment and availability of new services.

When dynamic adaptation (DA) occurs at runtime, it has to be performed within given time bounds, and the resulting system must comply with the execution time established for the system.

Moreover, the resulting system must comply with the execution time established for the system as a whole. Each one of these requirements has to be addressed accordingly. For instance, preservation of data integrity requires a mechanism to verify that the integrity of the data is kept during and after the adaptation. Moreover, timeliness and performance related requirements can be addressed by maintaining the execution times within the desired time bounds.

While the possibility of dynamic service deployment and evolution offers an exciting range of application development opportunities, it also poses a number of complex engineering challenges. These challenges include detecting adaptation triggers, facilitating timely, dynamic service composition, predicting the temporal behaviour of unplanned service assemblies and preventing adverse feature interactions following dynamic service composition. Therefore, a dynamically adaptable service has to be able to identify triggers for adaptation, select and direct an adaptation strategy, and preserve desired QoS properties, while avoiding undesired behaviours during or after the adaptation.

A promising approach to achieving the required properties in Dynamically Adaptable (DA) services is the use of formal methods and techniques, which have been successfully applied for managing complexity and system development to ensure implementations of high quality. Formal methods provide the foundation to verify and validate the behaviour of adaptive programs, as well as, the architecture engineering processes that validate a program against desired functional properties. However, most previous efforts to validate dynamically adaptable services against desired functional properties impose high-verification costs as each adaptation must be separately modelled and verified, making these approaches extremely expensive (See Section II). The result is that building DA services is either expensive or limited to predefined adaptation scenarios. Even more, these approaches focus their analysis on the resulting program neglecting the analysis of the adaptation mechanism itself, that is, the Adaptation Manager (AM). We consider the adaptation mechanism to be of capital importance in the development of DA services, since it is responsible for monitoring compliance of adaptation to desired properties and functionalities. Consequently, this work endeavours to provide a formal model of the AM.

Current approaches to DA offer various mechanisms for handling adaptation, such as Generic Interceptors [1], DA with Aspect-Orientation [2], Dynamic Reconfiguration [3], Dynamic Linking of Components [4], and Model-Driven Development of DA Software [5]. However, a more generic framework to verify the adaptation mechanism against commonly accepted service properties is still needed.

Formal languages provide the underpinnings to describe and model software systems in a precise manner, and are fundamental for the level of analysis, validation and proof required for assuring adaptation compliance to specifica-

 ISBN: 978-1-61208-156-4

tions. In this paper, we propose a concrete model for the AM, which provides the groundwork for a conceptual model. Moreover, the mathematical foundation selected for this study facilitates its validation against predefined QoS properties, such as, availability and responsiveness.

This article is organized as follows: Section II discusses related work. Later on, Section III provides an overview of the language chosen to describe the model. Section IV introduces our formal model and Section V discusses its properties.

Finally, in Section VI we draw some conclusions and introduce future work.

II. State of the Art

We introduce related work in two groups. First, those approaches specifically related to DA, second, formal approaches that are applied to analyse similar families of problems as the ones presented here.

Dynamic Adaptation

Mckinley and Geihs ([6], [7] and [8]) present an overview of DA and its constituents, however they do not advance a formal model or proposal to explore DA, which is the aims of our work. Similarly to the elements of DA we identified, Segarra and André describe a similar model to ours with components that can be customized for different applications, a component in their framework can be provided with a controller which performs the adaptation depending on execution conditions [9]. In our proposal we define one controller, the adaptation manager, that gathers information from supporting services such as timing and execution evaluation in order to perform adaptations. Work has also been carried out to map BPEL to Process Algebras as Ferrara [10], to Pi-calculus as Abouzaid [11], and to Petri Nets as Ouyang et al. [12].

Formal approaches to DA

The work of Laneve and Zavattaro [13] on web services advances an extension to the π-calculus with a transaction construct, the calculus webπ. This model supports time and asynchrony. However it remains at a more abstract level and is not applied to dynamic adaptation. Ferrara [10] relies on process algebra to design and verify web services, this work also allows to verify temporal logic properties as well as behavioural equivalence between services. Compared to this work, our attempt is more general and is directed at the study of dynamic adaptation. Finally our proposal is aimed at identifying a formal service-oriented language for modelling dynamic adaptation, rather than advancing techniques for formal verification of web services or services as in the work of Ferrara. Mori and Kita [14] explore the use of genetic algorithms to dynamic environments and offer a survey on problems of adaptation to dynamic environments. The work of ter Beek at al. [15] reviews service composition approaches with respect to a selection of service composition characteristics and helps to underscore the value of formal methods for service analysis at design, specially service composition. The authors present a valuable analysis of formal approaches to service composition and elaborate a useful comparison. Amano and Watanabe [16] explore mechanisms to check consistency.

This work ventures at providing a concrete model of an AM on top of which a more abstract one can be built. Furthermore, the streamlined relation between Cows and the model checker CMC permits us to realize a formal analysis of the concrete model we introduce in Section IV.

III. The Service-oriented Language COWS

As a first step towards developing a model of dynamic adaptation, we explored a number of languages in [17]. Current service-oriented formal languages like PiDuce [18], SOCK/JOLIE[19], COWS[20], KLAIM [21], and SCC [22] offer each a different range of possibilities to model DA. Each of these languages has an underlying process algebra and constructs that support definition of services and expressing substitution and deactivation processes. The main method in PiDuce to model service substitution is through virtual machines while, in the case of COWS, it is modelled by delimited receiving and killing activities handled with its process calculus. JOLIE and PiDuce offer no deactivation process. After reviewing these languages, we concluded that the best fit language for modeling the runtime dynamic adaptation problem is COWS. Comparing the characteristics of the languages selected only COWS provides constructs for timing analysis. Timeliness was not the only deciding criteria, considering that0 adaptation of services only via channel renaming is sufficient to achieve DA is questionable, as is the case of PiDUce. In this regards, the composition mechanisms of SOCK/JOLIE are more adequate, yet again with no possibility to evaluate timeliness. The choice is clear and well founded.

The calculus for orchestration of web services (COWS) [23] is a new process calculus similar to the Business Process Execution Language (WS-BPEL) [24]. However, contrary to WS-BPEL, COWS provides a formally specified distributed machine. COWS is a process calculus that allows to specify service-oriented applications, as well as, modelling dynamic behaviour. COWS also provides support for the development of tools to check that a given service composition follows desired correctness properties and unexpected behaviours are avoided.

COWS has been extended with timing elements [20] which facilitate adoption of the language for modelling services with timing requirements. Therefore, this language represents an important line of research in service-oriented computing (SOC). The motivation of the authors to develop COWS is that most formalisms “do not model the different aspects of currently available SOC technologies in their completeness” [20].

 ISBN: 978-1-61208-156-4

Services are structured activities built from basic activities, such as the empty, kill, invoke, receive, and wait. Services are composed by means of prefixing, choice, and parallel composition. Constructs as protection, delimitation, and replication are also provided. The language is parameterised by a set of expressions. Partner names and operation names can be combined to designate communication endpoints, written "$p \bullet o$", where p is a partner and o an operation. These represent activities of type receive "$p \bullet o?$" or invoke $p \bullet o!$".

Timed activities are frequently exploited in service-oriented computing and are needed to model time outs. Passing of time is modelled synchronously for services deployed on the same 'service engine' and asynchronously otherwise. In this language, time passes synchronously for all services in parallel, given that services run on the same service engine. An important aspect in dynamic adaptation is represented by timing constraints for either the execution of a new service or the execution of the system as a whole, therefore, elements for modelling time are needed

COWS [25] provides a proxy mechanism to add compensation handling in existing services. This means that it may add behaviour to existing services in the system by appending a compensatory service. A compensatory service is one that adds to the base service and achieves some corrective or compensatory functionality. We can also avail of sequential composition for adapting an existing protocol to the new one.

In COWS basic actions are *durationless* and the passing of time is modelled by explicit actions. Imperative and orchestration constructs support specification of assignment in variables, conditional choice and sequential composition. Conditional choice and sequential composition of services can be used to achieve dynamic adaptation by composing services with existing ones. The language also defines the concept of service engines, where each engine has its own clock which is synchronised with the clock of other parallel engines. All instances of a service run within the same engine. Moreover, time elapses between each evaluation of expressions and these evaluations are instantaneous. Only the time construct $\oplus_{argument}$ consumes time units. Time elapses while waiting for invoke or receive activities and the argument for wait-activities "$\oplus$" is set to the current stand. Parallel composition of engines in this language is given by sequential composition. Moreover, COWS provides a deactivation activity that forces termination of all unprotected parallel activities. Sensitive code can be protected from killing by placing it into a "protection". COWS computational entities are called *services* [23]. Services in COWS do not have interfaces since communication is realised through message passing among services, which are structured activities built from basic activities.

Finally, COWS can model several typical aspects of web services technologies, for instance, multiple start activities, receive conflicts, routing of correlated messages, service instances and interactions among them [20].

IV. A Model for Dynamic Adaptation

In previous work [26], we identify several techniques to achieve DA. Most of the current DA techniques are static, and more importantly, the flexibility of adaptation or the level at which adaptations are achieved, is in most cases limited. An area of opportunity we identified, is the need for a formal approach in DA. A more formal approach to DA allows system designers and architects to perform an analysis of the adaptive elements in the system. This is the motivation behind our work.

This section presents a two-step process for establishing our DA model.

1) A scenario explaining a dynamic adaptation problem
2) Design of the model in a formal language suitable for verification

Scenario

We assume an all-electronic toll system where tolls are collected with the use of a transponder mounted on the windscreen of each vehicle. This is a straightforward case, however, we further assume that the car travels along a number of borders and jurisdictions. In each jurisdiction the toll system may actually differ from the previous one. We propose an adaptable electronic toll collection system on the client (vehicle) side. This system would allow the owner to install only one toll system that interacts with the toll booth at each contact point. In this scenario we assume the car receives a welcoming signal every time it approaches a tollbooth. This is illustrated in Figure 1. In this diagram we restrict the representation to the sequence of events in the interaction between three interacting roles `Toll booth`, `Vechicle Communication`, and `Toll booth ETS service`. As we can see, the Toll booth sends a welcome signal to every vehicle approaching it. On receipt of the signal, the vehicle's communication service requests the type of payment protocol from the toll booth. The Vehicle identifies the protocol and evaluates its compatibility to the one currently installed. In case it is not compatible, the Toll booth is notified and requests a new protocol for the car to be sent together with an estimate of the car's arrival time to the toll collection area. This estimate is used as the upper time bound for the subsequent adaptation of the vehicle's Electronic Toll Payment Service.

In the following we present a model for DA suitable for validation against desirable attributes of services and service-oriented computing applications. According to this, a service should be: Available, Reliable, and Responsive.

Model

The model of DA is introduced in Figure 2. Here we assume a runtime monitor in the software composite that activates the AM based on predefined triggers.

 ISBN: 978-1-61208-156-4

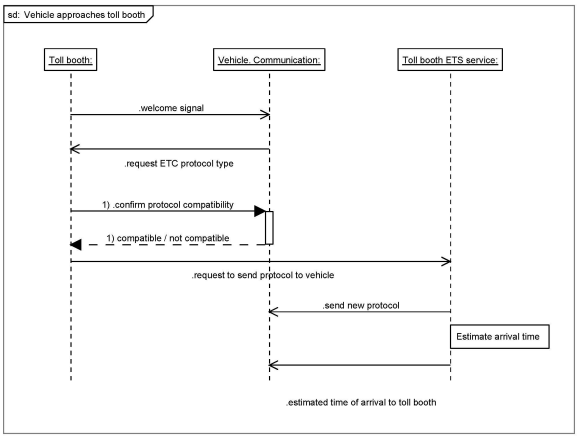


Figure 1. Sequence Diagram Tollbooth

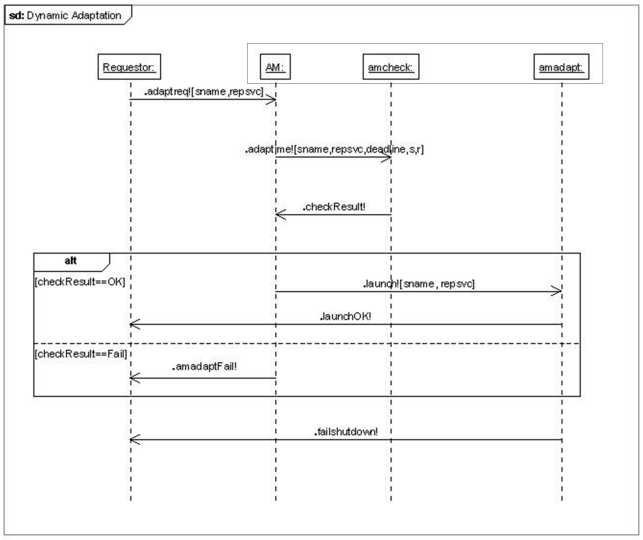


Figure 2. Sequence Diagram Adaptation Process

```
                                                                    1
let
adaptManager(service) =                                             3
    * [X][Y][Z][XX][YY]
    service.create?<X,Y,Z,XX>.p.adaptime!<X,Y,Z,XX>                 5
  | [X][Y][Z][XX]
    ser.checkOK?<X,Y,Z,XX>.q.exectime!<Z,XX>                        7
  | ser.checkFail?<>. ser.launchFail!<repsvc>
  | ser.checkFail2?<>. ser.launchFail!<repsvc>                      9
requestor() =
    serv.create!<0,4,10,60>                                         11
  | ser.checkOK2?<>.amadapt.launchOK!<> |
    (amadapt.launchOK?<>.s.signalOK!<>                              13
    + ser.launchFail?<repsvc>.s.signalFail!<>
    + ser.launchFailx?<>.s.signalFail!<>)                           15
in
adaptManager(serv)                                                  17
| requestor()
| * amcheck()                                                       19
| amcheck2()
| s.signalFail?<>.nil                                              21
| s.signalOK?<>.nil
end                                                                 23
```

Listing 1
ADAPTATION MANAGER, REQUESTOR AND MAIN SERVICE

```
Amcheck_gt_deadline(X) =
  (ser.checkFail!<>)                                                2
Amcheck_le_deadline(X,Y,Z,XX) =
  (ser.checkOK!<X,Y,Z,XX>)                                          4
Amcheck_gt_deadline2(X) =
   (ser.checkFail2!<>)                                              6
  | memory.assert?<X>.nil
Amcheck_le_deadline2(X) =                                           8
  (ser.checkOK2!<>)
amcheck()=                                                          10
    [X][Y][Z][XX]
    p.adaptime?<X,Y,Z,XX>.                                          12
    [i#]
    (i.selectgreater!<X gt Y> |                                     14
      (i.selectgreater?<true>.
       Amcheck_gt_deadline(X) +                                     16
       i.selectgreater?<false>.
       Amcheck_le_deadline(X,Y,Z,XX)                                18
      )
    )                                                               20
```

Listing 2
ADAPTATION-TIME CHECK

The adaptation process initiates by sending a signal (`adaptreq(sname,repsvc)!`) from the `Requestor` to the `AM`. This is represented in Listing 1 line 11. The AM further sends the timeliness parameters to evaluate the plausibility of the adaptation in view of the time constraints, as illustrated by the second message `adaptime(sname,repsvc,deadline,s,r)`. Once the object "amcheck" receives this signal, the timeliness conditions are evaluated and the result returned to the AM. The flow for the adaptation proceeds, when the service `amcheck` is called with the time estimations in Listing 1 line 7. In this point, a procedure to compare the estimated time of the adaptation against the predefined upper bound is called as expressed in Listing 2 lines 12 to 18. Next, in case the adaptation can be fulfilled within the defined time bounds, a second condition is evaluated. This is represented in Listing 3 lines 3 to 10. This evaluation verifies that the adaptation preserves the overall execution time of the service. In case both conditions are uphold, the service `amadapt` is executed. After execution, this service sends the signal `cheqOK2!` (Listing 1 line 12) to the requestor, which in turn notifies the user on a successful adaptation by sending the signal `launchOK`. Otherwise, the AM responds `signalFail`.

The model focuses on two timeliness conditions, adaptation time and execution time. However, other conditions can be analyzed by the AM in the same manner. For instance, verifying that the reconfigured service is in a quiescent state before performing any changes in order to avoid interaction and coordination conflicts. In the following section, we describe the verification process that the model grants.

 ISBN: 978-1-61208-156-4

```
amcheck2()=
    [X][Y]
    q.exectime?<X,Y>.
    [i#]
    [K]
    (i.selectgreater!<X gt Y> |
      (i.selectgreater?<true>.
       Amcheck_gt_deadline2(X) +
       i.selectgreater?<false>.
       Amcheck_le_deadline2(X)
      )
     )
```

Listing 3
EXECUTION-TIME CHECK

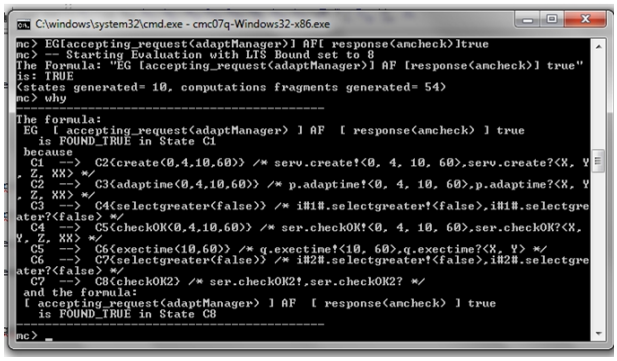
V. ANALYSIS OF THE MODEL

Dynamic software architectures allow us to build dynamically adaptable systems by supporting the modification of a system's architecture at runtime. Possible modifications include structural adaptations that change the system configuration graph, for example, the creation of new and the deletion of existing components and connectors, as well as functional adaptations where components are replaced. Further, even more challenging changes are those that modify the behaviour of components and ultimately services.

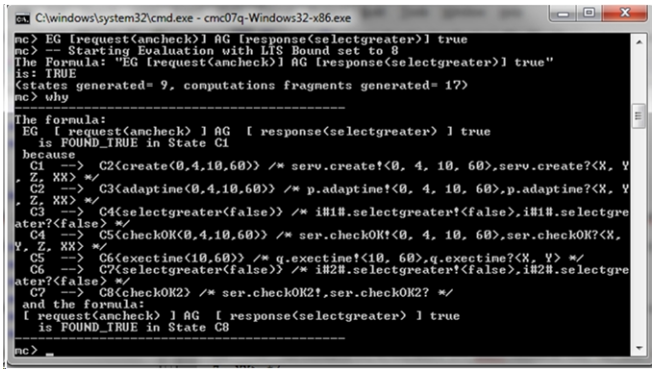
To achieve reliable dynamic adaptable services we need to evaluate service modifications against a number of quality of service properties. Services must comply as a minimum with the following three properties. The first one, Responsiveness, means that the service always guarantees an answer to every received service request, unless the user cancels. The second property, Availability, requires that the service is always capable to accept a request. Finally, the third property, Reliability means that the service request can always succeed.

In order to evaluate a system against the properties mentioned above, we first express these properties and the model in a language that supports a formal analysis. This analysis should be performed preferably in an automated manner. Cows, the language we selected, possesses this quality. In the following we introduce the selected model checker, which is CMC. Cows grants straightforward coupling with CMC. Afterwards, we explore the model for reliability and responsiveness properties with the model checker.

For reasons of space, we present only the model checking results for Responsiveness and Reliability. To explore the first property, Responsiveness, we define a formula with a universal quantifier on the execution of `amcheck` following a call to the AM specified as an existential quantifier to the AM, as shown in Listing 4. A run of the model checker on this condition returns `true` and no counter-example as shown in Figure 3(a). Verification of the model against Availability follows a similar pattern. In general terms, the User identifies the service name of interest, in this case the AM and specifies the related logical condition as a universal quantifier over the request for the adaptation manager. The AM initial call is found in Listing 1, which is input to the model checker and the conditions is tested.



(a) Responsiveness Check with CMC



(b) Reliability Check with CMC

Figure 3. CMC Runs

```
AG[request(adaptManager)]true
EG[accepting_request(adaptManager)] AF[ response(amcheck)] true
```

Listing 4
RESPONSIVENESS

```
EG[request(requestor)] EG[response(launchOK)] EG[response(launchFail)]true
```

Listing 5
RELIABILITY

The formula for Reliability is built by the combination of existential quantifiers over the services `requestor`, `launchOK` and `launchFail` in Listing 5. The model-checker renders no counter-example, but `True` as shown in Figure 3(b).

VI. CONCLUSIONS

Real time DA is an area of research that poses new challenges to software development, where the goal is to produce a system capable of adapting to changing conditions in the operational environment. Additional demanding goals related to DA are the integration of new services as these become available, or coping with reconfiguration issues, all this

at runtime and under time constraints. DA has been proposed to provide solutions for these challenges. A methodology for the study of DA is still an open question. Formal methods have been in use for a long time in the computer science community and a number of new approaches and formal languages are available. We suggest that modelling DA with a formal language can provide precise answers to most of the existing questions and grant a better understanding. As a consequence, in this work, we recommend the use of the formal language COWS to model DA, introduced a first formal model, and assessed it by checking against three widely accepted service properties: Responsiveness, Availability and Reliability, with the model checker CMC.

ACKNOWLEDGMENT

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme

REFERENCES

[1] S. M. Sadjadi and P. K. McKinley, "Act: An adaptive corba template to support unanticipated adaptation," *icdcs*, vol. 00, pp. 74–83, 2004.

[2] Z. Yang, B. H. C. Cheng, R. E. K. Stirewalt, J. Sowell, S. M. Sadjadi, and P. K. McKinley, "An aspect-oriented approach to dynamic adaptation," in *WOSS '02: Proceedings of the first workshop on Self-healing systems*. New York, NY, USA: ACM, 2002, pp. 85–92.

[3] M.-C. Pellegrini and M. Riveill, "Component management in a dynamic architecture," *J. Supercomput.*, vol. 24, no. 2, pp. 151–159, 2003.

[4] C. Escoffier and R. S. Hall, "Dynamically adaptable applications with iPOJO service components," in *Software Composition*, ser. Lecture Notes in Computer Science, M. Lumpe and W. Vanderperren, Eds., vol. 4829. Springer, 2007, pp. 113–128.

[5] J. Zhang and B. H. C. Cheng, "Model-based development of dynamically adaptive software," in *ICSE '06: Proceeding of the 28th international conference on Software engineering*. New York, NY, USA: ACM, 2006, pp. 371–380.

[6] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. C. Cheng, "A taxonomy of compositional adaptation," Dept. Computer Science and Engineering, Michigan State University, Tech. Rep. MSU-CSE-04-17, 2004.

[7] P. K. McKinley, S. M. Sadjadi, E. P. Kasten, and B. H. Cheng, "Composing adaptive software," *Computer*, vol. 37, no. 7, pp. 56–64, 2004.

[8] K. Geihs, "Selbst-adaptive software," *Informatik-Spektrum*, 2007, 0170-6012 (Print) 1432-122X (Online).

[9] M.-T. Segarra and F. André, "A framework for dynamic adaptation in wireless environments," in *TOOLS '00: Proceedings of the Technology of Object-Oriented Languages and Systems (TOOLS 33)*. Washington, DC, USA: IEEE Computer Society, 2000, p. 336.

[10] A. Ferrara, "Web services: a process algebra approach," in *ICSOC '04: Proceedings of the 2nd International Conference on Service Oriented Computing*. New York, NY, USA: ACM, 2004, pp. 242–251.

[11] F. Abouzaid, "A mapping from pi-calculus into bpel," in *Proceeding of the 2006 conference on Leading the Web in Concurrent Engineering*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2006, pp. 235–242.

[12] C. Ouyang, E. Verbeek, W. M. P. van der Aalst, S. Breutel, M. Dumas, and A. H. M. ter Hofstede, "Formal semantics and analysis of control flow in ws-bpel," *Sci. Comput. Program.*, vol. 67, no. 2-3, pp. 162–198, 2007.

[13] C. Laneve and G. Zavattaro, "Foundations of web transactions." Springer, 2005, pp. 282–298.

[14] K. H. Mori Naoki, "Genetic algorithms for adaptation to dynamic environments - a survey," in *26th Annual Conference of the IEEE Electronics Society IECON 2000*, I. P. I. E. Conference), Ed., 2000.

[15] A. B. Maurice H. ter Beek and S. Gnesi, "Formal methods for service composition," *Annals of Mathematics, Computing & Teleinformatics*, vol. 1, 5, pp. 1–10, 2007.

[16] W. T. Amano Noriki, "Towards constructing component-based software systems with safe dynamic adaptability," in *International Workshop on Principles of Software Evolution (IWPSE)*, 2001.

[17] J. Fox and S. Clarke, "An analysis of formal languages for dynamic adaptation," in *International Conference on Engineering of Complex Computer Systems (ICECCS 2010)*, March 2010, pp. 3 – 13.

[18] S. Carpineti, C. Laneve, and L. Padovani, "PiDuce – a project for experimenting web services technologies," *Science of Computer Programming*, vol. 74, no. 10, pp. 777 – 811, 2009.

[19] C. Guidi, R. Lucchi, R. Gorrieri, N. Busi, and G. Zavattaro, "SOCK : A calculus for service oriented computing," *Proceedings of Service-Oriented Computing ICSOC 2006*, vol. 4294/2006, pp. 327–338, 2006.

[20] A. Lapadula, R. Pugliese, and F. Tiezzi, "Cows: A timed service-oriented calculus," in *Proc. of 4th International Colloquium on Theoretical Aspects of Computing (ICTAC'07)*, ser. Lecture Notes in Computer Science, vol. 4711. Springer, 2007, pp. 275–290.

[21] L. Bettini, V. Bono, R. D. Nicola, G. Ferrari, D. Gorla, M. Loreti, E. Moggi, R. Pugliese, E. Tuosto, and B. Venneri, "The KLAIM Project: Theory and Practice," in *Global Computing: Programming Environments, Languages, Security and Analysis of Systems*, ser. LNCS, C. Priami, Ed., no. 2874. Springer, 2003, pp. 88–150.

[22] M. Boreale, R. Bruni, L. Caires, R. D. Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. T. Vasconcelos, and G. Zavattaro, "SCC: A service centered calculus," in *Web Services and Formal Methods*, ser. Lecture Notes in Computer Science, M. Bravetti, M. Núñez, and G. Zavattaro, Eds., vol. 4184. Springer, 2006, pp. 38–57.

[23] A. Lapadula, R. Pugliese, and F. Tiezzi, "A Calculus for Orchestration of Web Services," in *Proc. of 16th European Symposium on Programming (ESOP'07)*, ser. Lecture Notes in Computer Science, vol. 4421. Springer, 2007, pp. 33–47.

[24] "Business Process Execution Language for Web Services (BPEL4WS)," http://xml.coverpages.org/bpel4ws.html, last accessed 01.09.2011.

[25] "Calculus for Orchestration of Web Services (COWS)," http://rap.dsi.unifi.it/cows/, last accessed 01.09.2011.

[26] J. Fox and S. Clarke, "Exploring approaches to dynamic adaptation," in *Proceedings of the 3rd International DiscCoTec Workshop on Middleware-Application Interaction*. New York, NY, USA: ACM, 2009, pp. 19–24.

 ISBN: 978-1-61208-156-4

Real-Time Transfer and Evaluation of Activity Recognition Capabilities in an Opportunistic System

Marc Kurz, Gerold Hölzl, Alois Ferscha
Johannes Kepler University Linz
Institute for Pervasive Computing
Linz, Austria
{*kurz, hoelzl, ferscha*}*@pervasive.jku.at*

Alberto Calatroni, Daniel Roggen, Gerhard Tröster
ETH Zürich
Wearable Computing Laboratory
Zürich, Switzerland
{*alberto.calatroni, daniel.roggen, troester*}*@ife.ee.ethz.ch*

***Abstract*—This paper describes and evaluates the challenging feature of an opportunistic activity recognition system to train a newly discovered sensor with the available sensing devices to recognize activities at runtime. The term "opportunistic" means that the system does not operate with a fixed set of sensor devices, but uses and configures the currently available sensors that just happen to be available. Therefore, the paper presents a reference implementation of an opportunistic system, referred to as *OPPORTUNITY Framework*, and demonstrates the transfer of recognition capabilities from a fused multi-sensor ensemble to an untrained sensing device within the system in a real-world setup. Main contribution of the paper is the evaluation of the approach by describing an experimental setup and presenting results in terms of accuracy and recognition rate from the machine-learning perspective as well as from the framework and system perspective by comparing predicted classes from the teaching sensor set and the newly trained sensor to obtain QoS parameters.**

***Keywords*-Activity and Context Recognition; Opportunistic Sensing; Senor Networks.**

I. INTRODUCTION

Activity and context recognition systems utilize sensing devices that are available in the environment, on objects, or on persons to sense the world in terms of inferring activities. Traditionally, machine learning technologies that interpret the sensor datastreams are trained in an offline mode, at the design time of the system [1]. In contrast to that, an opportunistic system does not specify the set of required sensor systems a priori, instead it utilizes sensor nodes that just happen to be available to recognize the person's activities. Another major characteristic of an opportunistic system is the fact that recognition goals are defined dynamically at runtime by an application or a user, and the available sensing devices are configured to an *ensemble*, which is the set of accessible sensors that are best suited to execute this goal [2]. The type and modality of the involved sensor systems that are utilized to execute a recognition goal cannot be pre-defined as sensors are used that just happen to be available. Therefore, the system has to handle physical, logical, and other types of sensors [3] in order to execute a recognition goal. Given these definitions, some characteristic features and application cases of an opportunistic activity recognition system can be identified, like: (i) *sensor appears*, (ii) *sensor disappears*, (iii) *sensor reappears*, (iv) *sensor delivers reduced-quality data* and (v) *sensors are trained for an active recognition goal at runtime* [4].

This paper presents the feature that a newly appeared sensor (*Learner*) can be trained with the existing ones that are executing a recognition goal (*Teacher(s)*) by providing the predicted activity class to incrementally train the new sensor and calculate QoS parameters [5] on the fly to estimate to what extent the learner will be able to contribute to a future, similar recognition goal. Therefore, we use the *OPPORTUNITY Framework*, a reference implementation of an opportunistic activity recognition system (see [3] [4] [5]). By utilizing a *system-supervised learning* approach on the new sensor node [6] we compare the predicted label of the teacher to the label predicted by the learner to calculate a *degree of fulfillment (DoF)* [5] metric that indicates to what extent a sensor can fulfill a certain recognition goal. This information together with the dynamically obtained machine learning parameters (e.g., classifier model) is stored persistently in the sensor's self-description. We present and evaluate the approach by operating the OPPORTUNITY Framework in a real-world scenario with four body-mounted sensor devices that deliver triaxial accelerometer data. We transfer the capability to recognize the locomotion of a person (i.e., *WALK*, *SIT*, *STAND*, *LIE*) from a 3-sensor ensemble to a single sensor and compare the teacher with the learner output class to evaluate the approach. This is radically different from standard settings using offline training, since we cannot gather groundtruth labels here to assess the performance of the learner. We instead have to rely only on the teacher-learner comparison.

The remainder of the paper is structured as follows: Section 2 provides an overview on related work. Section 3 presents a description of the technical details and the realization of the sensor learning approach. Sections 4 and 5 describe an experimental setup with on-body sensors and the results of the training. Section 5 closes with a conclusion and an outlook.

 ISBN: 978-1-61208-156-4

II. Related Work

Traditional activity recognition systems have to define the classes that shall be recognized, the sensors and their operating characteristics at design time of the system. The activity recognition chains that process sensor signals to infer activities involve different steps: datastream preprocessing, feature extraction, classification and multi-sensor fusion. The classification step (mapping feature vectors to a defined set of output classes) involves offline-trained machine learning algorithms [1] [7] [8].

As opportunistic sensing and opportunistic activity recognition draw from the characteristic to use sensor nodes that just happen to be available to execute a dynamically-stated recognition goal (see [2] [3] [4] [5]), an approach to make sensors ready to recognize activities on the fly at runtime of the system is necessary. This transferring of recognition capabilities from one (or more) sensor(s) to another sensor can be done on the classifier level, where the model is transferred directly from one sensor node to another [6]. This approach suffers from the problem that both nodes have to operate on the same feature space, which limits the approach. Another way is to provide the feature space independent classes to the learner, which can incrementally train the baseline machine learning technologies [6]. In [9] the authors showed the transfer of activity recognition capabilities from one smart home to multiple different systems operating in the same domain. Again, the transfer relies on a common feature space. The transfer of capabilities across multiple feature spaces is also shown in [10].

This paper takes on the approach that is presented in [6] and shows that the transfer of recognition capabilities in an opportunistic environment is possible in a real-world scenario. As no ground truth is usually available, an estimate of the QoS for the learner has to be calculated. This is done in form of a DoF that is calculated by comparing the predicted class from the teacher with the predicted class from the learner and stored as part of the sensor self-description. Details of these self-descriptions are presented in [5]. This paper extends the *system-supervised learning* approach that operates independently from the feature spaces of the teachers and learners as described in [6]. There, the approach is described and evaluated on a rich dataset [11] and by using Nearest Class Center (NCC), k-Nearest Neighbors and SVM classifiers with the advantage to have a groundtruth available. Main contribution of this paper is the application of the approach in the OPPORTUNITY Framework, and the empirical calculation of the accuracy in form of the DoF during the training phase at runtime.

III. Technical Details and Realization

This Section provides technical details of the OPPORTUNITY Framework, the applied sensor self-description concept, and how the transfer learning approach is realized within the framework.

A. The OPPORTUNITY Framework

A reference implementation of an opportunistic activity recognition system (the *OPPORTUNITY Framework*) was developed and is used within this paper. The framework is written in Java/OSGi and is a step towards a ready-to-use middleware for building opportunistic activity recognition applications in different domains [3] [4] [5]. An opportunistic system utilizes sensors in a way to configure the best set of sensors according to a recognition goal. If we assume that the set is not static, then our system needs to react on changes in the sensing infrastructure (e.g., a node might disconnect when running out of power). The following more or less challenging features can be identified that characterize such an opportunistic system in terms of (self-) adaptation:

(i) *Sensor appears*: a new sensor joins the sensing infrastructure. If the sensor is already capable of contributing to the recognition goal, the system has to assess whether or not the sensor is able to increase the overall ensemble's contribution to the recognition goal. If the sensor is still untrained, it can be trained by the other sensor(s). The system gets the knowledge of the sensor's capabilities by parsing its self-description.

(ii) *Sensor disappears*: when a sensor disconnects, the system reaction depends on whether the sensor was or was not active in a configured ensemble. In case the sensor was not active, the current sensor ensemble does not have to be reconfigured. In the other case, a reconfiguration could be needed.

(iii) *Sensor reappears*: same as *sensor appears*, but the system already knows the sensor's capabilities as it has parsed its self-description on previous connections.

(iv) *Sensor delivers faulty data*: when a sensor is shifted or broken it could be that it still delivers (reduced-quality) data. The system has to recognize this and reduce the trust indicator metric of the sensor [5] and - if necessary - reconfigure the sensing ensemble.

(v) *Transfer of recognition capabilities to a sensor*: this is the main focus of this paper. A newly connected sensor has to be trained by the other sensors in terms of recognizing activities and an estimation needs to be provided for the achieved accuracy.

(vi) *Ensemble configuration at runtime*: this is a key aspect in an opportunistic system. Whenever a recognition goal is stated to the system, the set of sensors that are best suited to execute this goal are configured. This process [5] has to be executed whenever something happens in the sensing infrastructure.

The next Section III-B explains the concept of sensor self-description and how this can be used to perform real-time transfer learning, and III-C describes how transfer learning is implemented in the OPPORTUNITY Framework and how its performance can be measured in an online setting.

 ISBN: 978-1-61208-156-4

B. Sensor Self-Description

The sensor self-description is an important aspect in an opportunistic activity and context recognition system. It provides the technical details of the sensor to the system and builds the connection point between the high-level framework features and the machine learning technologies on the lower levels. We have split the self-description into two parts: (i) the technical description that holds the physical characteristics as well as technical aspects of a sensor (e.g., power requirements, communication interface, update rate, size, weight, ...), and (ii) the dynamic description that lists the sensor's capabilities according to recognition goals [5]. Both parts of the sensor self-description follow the OpenGIS SensorML specification, an XML standard definition. The concept of *ExperienceItems* defines a complete recognition chain of a sensor together with the required signal processing and machine learning techniques (feature extraction, classification, fusion) to recognize activity classes. ExperienceItems are part of the dynamic self-description and there can be multiple items describing multiple activity classes or recognition chains [5]. These ExperienceItems and the defined methodologies can be invoked and configured by the activity recognition system at runtime on demand. The DoF defines to what extent a sensor together with the recognition chain can execute a recognition goal. Figure 1 shows a clipping of an ExperienceItem where the DoF is defined for the activity class *WALK*. This means, that the sensor and the corresponding machine learning techniques as defined in this very ExperienceItem can execute the recognition goal *WALK* with a DoF of 0.75. This degree of fulfillment metric reflects the expected accuracy in the recognition of a certain activity and can be generally calculated in two ways:

(i) *Statically by using a labeled groundtruth for learning*: normally a system is trained by using a groundtruth that defines the sensor signal and patterns and the corresponding activity classes. The classification mechanisms are therefore trained with the labeled groundtruth data to autonomously detect significant and similar patterns in the datastream. This approach presumes the initial knowledge of the used sensors, their modalities, the feature space, the exact position and location and the activity classes that shall be recognized (and of course a groundtruth of appropriate size/length).

(ii) *Dynamically by using transfer learning*: this method is the core contribution of the paper and will be described in detail in the following Section III-C.

To avoid the need for a labeled groundtruth and thus the training in offline mode, this paper presents and evaluates the approach of training new sensor nodes at runtime of the system and calculating the DoF by comparing the predicted class from the teacher with the predicted class from the learner. The next Section describes the approach of real-time transfer learning and the calculation of the DoF in detail.

```
...
  <swe:field name="labellist">
   <swe:DataRecord>
    <swe:field name="WALK">
     <swe:Text definition="dof">
      <swe:value>0.75</swe:value>
     </swe:Text>
    </swe:field>
   </swe:DataRecord>
  </swe:field>
...
```

Figure 1. A clipping of an ExperienceItem as part of a sensor self-description together with the DoF metric [5].

C. Application of Transfer Learning

When a new sensor appears, the system parses its self-description to check whether it can contribute to a running execution of a recognition goal, or if not, whether it can be regarded as a learner candidate for one of the goals that are currently active. A learner candidate has to be defined so by a human expert. For example, one could define an accelerometer on the shoe as possible sensor to detect the modes of locomotion. Learner candidates have in their self-description a feature extraction and a classification method, which form a default ExperienceItem template. Their initial DoF is set manually to "0.0". Thus, the system recognizes a learner candidate for a recognition goal when a sensor appears in the sensing environment with an initial zero DoF value (for this very goal). This means that the sensor can be picked for learning, and as soon as an ensemble that executes the recognition goal is configured and active, the learning process is initiated. The predicted classes from the teacher are used by the learner to generate the classifier model (persistently stored in form of a JSON file, see Figure 2 for an example) on the fly by assigning the activity class to the extracted features from the datastream.

The calculation of the effective learner DoF is not a trivial task, since groundtruth information is not available at runtime. For the calculation of the DoF, we can only rely on comparisons between teacher and learner, taking into account that the teacher does not provide a perfect groundtruth. We calculate the agreement rate using the cumulative moving average, which is a statistical method to analyze time series data [12]. In detail, the following values and variables are used to calculate the DoF of the learner at runtime:

- DoF_T = DoF from the teacher (known from its ExperienceItem).
- DoF_L = DoF from the learner that has to be calculated on the fly.
- n = count of predicted and compared activity classes.
- ϑ = degree of teacher-learner agreement during the training phase, defined as the percentage of instances where teacher and learner output the same activity class label among the total number of examined instances.
- $[0|1]$ = false and true, indicates the match (true) or

mismatch (false) between the class labels predicted by Teacher and Learner.

DoF_L is calculated as follows as soon as the learning process ends:

$$DoF_L = DoF_T * \vartheta, \quad (1)$$

which means that as soon as the learning process and the comparison of the classes predicted by the learner and teacher are over, the DoF of the learner is calculated by multiplying the DoF of the teacher with the cumulative average of the agreement rate. This multiplication has the goal to rescale the calculated DoF in cases where we have a high agreement between teacher and learner (ϑ close to 1) and an imperfect teacher. In these cases, in fact, we have high agreement between a learner and a wrong teacher, which does not mean at all that the learner is fulfilling the recognition goal properly. As a side effect, the estimated DoF_L will always be smaller or equal to the DoF_T, which is indeed often the case, but the opposite can also happen, as can be seen in a few cases in [6]. By the above calculation we are then accepting the condition $DoF_L \leq DoF_T$ and we are making a pessimistic estimate, which leaves us on the safe side when using the learner in following missions (the learner could perform even better than foreseen by the system).

The end of the learning process in our version of the framework is reached when either the teacher (or sensors within the teacher ensemble) or the learner is disconnected. Alternative approaches based on a time lapse or upon convergence of the DoF will be investigated in future work. The cumulative moving average of the agreement rate is calculated at runtime as follows:

$$\vartheta_{n+1} = \frac{\vartheta_n * n + [0|1]}{n+1}, \quad (2)$$

where n is incremented by one after each iteration. The number that has to be added to the numerator, after having multiplied ϑ with n, depends whether the class predicted by the teacher agrees or disagrees with the one predicted by the learner. In this way, the cumulative moving average of agreeing labels can be calculated over time, building the running accuracy for the learner measured with respect to the teacher. When the training phase is finished, this relative accuracy (ϑ) is multiplied by the DoF of the teacher to get an estimation for the DoF. This value and the newly built classifier models (in form of a JSON file) are written back to the sensor's self-description, where they complete the usual information about the used feature extraction and classifier.

IV. Experiment and Evaluation

We have set up a real-time scenario with body-worn sensors to test and evaluate the transfer of recognition capabilities for the modes of locomotion (*WALK, SIT, STAND, LIE*). We have picked a rather easy activity set that has to be recognized as the goal is not to work with sophisticated and highly complex activity classes, but to test to what extent rather simple and easy to recognize activities are transferrable. We used 4 sensors that delivered triaxial acceleration data. Two Intersense InertiaCube3 sensors were mounted on the right upper-/lower-arm (*motionjacket_RUA* and *motionjacket_RLA*, see Figure 3-1 and 3-2), one bluetooth accelerometer was mounted on the right knee of the subject (*btaccel_RKN*, see Figure 3-1 and 3-3), and one SunSPOT accelerometer sensor was attached to the right shoe (*sunspot_shoetoebox*, see Figure 3-1 and 3-3). The ensemble that was trained and configured to recognize the mode of locomotion activity classes consisted of the *motionjacket_RUA*, the *sunspot_shoetoebox*, and the *btaccel_RKN* sensors. The untrained sensor that was trained with the predicted classes from the teacher ensemble in our test setup was the *motionjacket_RLA* sensor. The setup of the three sensors in the ensemble was defined in an ExperienceItem:

- *Feature Extraction*: Mean/Variance for each sensor/recognition chain.
- *Classification*: NCC Classifier for each sensor/recognition chain.
- *Fusion*: Majority-Voting Fusion to combine the classification results for each sensor/recognition chain.
- *DoF*: 0.792 for *WALK, STAND, SIT, LIE* for the complete ensemble.

Below the picture of the test subject in Figure 3, also some sensor datastreams are shown. The *motionjacket_RLA* sensor was picked as learner as we set its DoF to detect modes of locomotion to zero. The predicted class from the ensemble was not only presented as system result, but also to the learner to incrementally train the classifier (NCC) model. The following Section V summarizes the results that were achieved within the experiment.

```
{
  "centroids" : [
    [-8.4965,-2.2945,2.5368,0.29659],
    [-9.2508,-2.3725,1.7388,1.3654],
    [-9.3933,-1.9461,1.0396,0.13042],
    [-4.4928,7.9204,-2.242,0.13042]
  ],
  "centroid_labels" : [3,1,5,4],
  "number_of_instances" : [1310,433,476,95],
  "cloud_size" : [
    [1.0241,0.89795,1.8205,0.2136],
    [0.32604,0.56007,0.71504,0.55195],
    [0.26634,0.57792,1.1387,0.11843],
    [1.1746,0.56141,1.5778,0.2331]
  ]
}
```

Figure 2. An example of a JSON file, which provides the configuration of a classifier used for activity recognition (the classifier model) [5].

V. Results

The learning phase in the experiment was 15 minutes long. So the teacher-ensemble presented its predicted label to

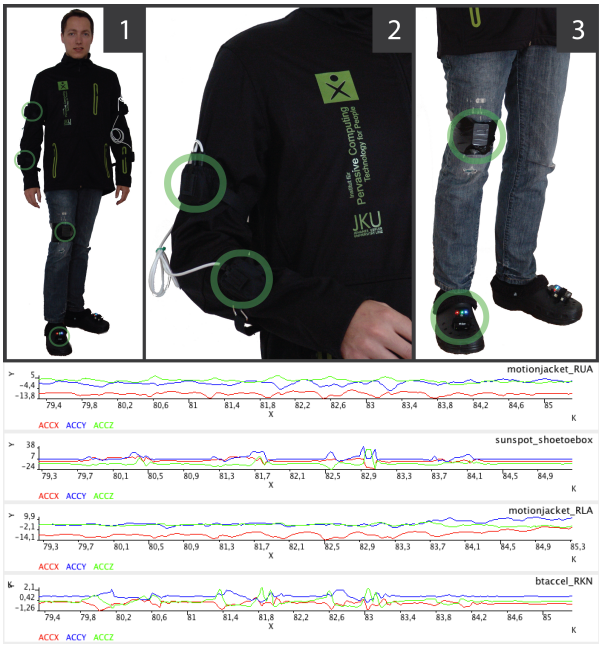


Figure 3. Experimental setup of the on-body sensors together with snapshots of the corresponding datastreams.

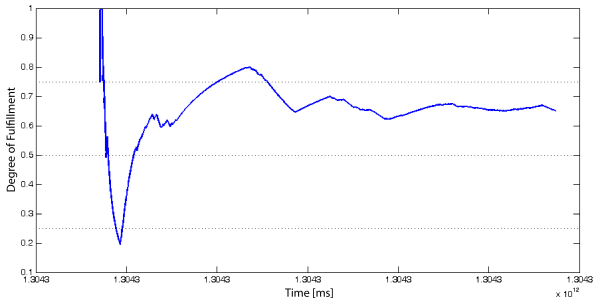


Figure 4. Evolution of ϑ of the *motionjacket_RLA* sensor during the training phase.

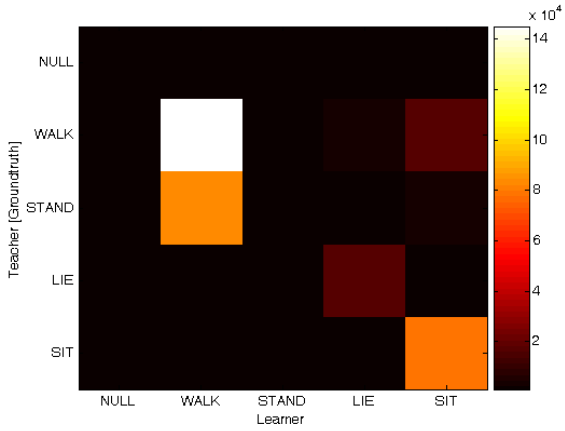


Figure 5. Visualization of the confusion matrix from the learner compared to the teacher (as groundtruth estimation).

the learner for executing the incremental training for 15 minutes, whereas the four activity classes (*WALK, STAND, SIT, LIE*) were executed by the subject approximately equally long but in a random way. That means the subject did not execute *WALK* for 3.75 min, followed by *STAND* sequentially. The activities occurred randomly but (approx.) equally long to ensure enough training samples per activity class. The ϑ value (which is the accuracy of the learner compared to the teacher during the training phase) over time during the real-time training process is shown in Figure 4. During the first few seconds, the value varied substantially (between 1 and 0.2), which can be explained by considering that the value has to settle over time until (i) the learner classifier model is well enough trained, and (ii) enough predicted classes are compared. Therefore, it is important to have a training phase that is long enough to have a good classifier model and a stable DoF calculation for the learner. In our experiment, the ϑ value settled to 0.65. This value cannot be seen as permanently stable and fixed, since it highly depends on (i) the number of observed activity classes, and (ii) the duration of the training phase. By multiplying the DoF_T value with ϑ we get a DoF for the learner of 0.515.

In Figure 5 the confusion matrix is shown that compares the predicted activity classes from the teacher with the predicted classes from the learner. There, the calculated DoF value from the learner is reflected, as we can notice a high occurrence of the false interpretation of the *WALK* and *STAND*/*SIT* activity classes. Altogether, during the 15 minutes training phase, we had approximately 350.000 comparisons from the teacher with the learner predicted class, which are contained in the confusion matrix. Table I summarizes the results that were achieved in the experiment.

Table I
SUMMARIZATION OF RESULTS

Duration of Training	15 minutes
Number of activity classes	4 (*WALK*, *STAND*, *SIT*, *LIE*)
DoF_T	0.792
Number of comparisons	approx. 350.000
ϑ (i.e., cumulative moving average)	0.65
DoF_L	$0.792 * 0.65 = 0.515$

Our approach of real-time training to realize self-adaption by online training new sensors and the persistent storage of this newly acquired knowledge at runtime works. The calculation of 0.515 as DoF is realistic and reflects the self-adaptation according to real-time transfer of recognition capabilities as not only correct predicted labels are transferred from the teacher to the learner but also wrongly classified activities.

 ISBN: 978-1-61208-156-4

VI. Conclusion and Future Work

We have presented the capability of an opportunistic activity and context recognition system to self-adapt in a way that untrained and newly appeared sensors can be trained by the existing ones. The recognition capabilities of the configured (teaching) ensemble are transferred in real-time to the learner candidates. During the learning process, the quality of the transferred recognition capabilities is quantified to get a stable and reasonable measurement of how good the learner can recognize the activity class. After DoF_L is estimated, this value is stored with the corresponding classifier model in the ExperienceItem of the sensor, to ensure persistency of the newly acquired knowledge. Our approach enables the autonomous transfer of recognition capabilities without relying on a labeled ground-truth to newly appeared sensors and therefore self-adapt to open-ended environments where sensors presumably not known at design time can be trained and used afterwards to extend the system.

There are many interesting aspects for future research. On one side, a probabilistic framework can be set up in order to provide a more accurate estimation of the learner accuracy given the rate of agreement between teacher and learner and by introducing all the possible knowledge about the teacher (like the confusion matrix). Different estimation techniques can be evaluated on existing datasets, where the groundtruth is not made available to the algorithms, but is used to assess how well the DoF can be estimated. Another interesting aspect is to investigate how to overcome the need for an expert to manually define learner candidates. This can be tackled by evaluating the suitability of a certain sensor due to other elements of its self-description, like the measured quantity (e.g., acceleration), and the placement. The end goal will be to design a planner, which will be able to select learner candidates, operate the transfer of capabilities only where it is meaningful and finally assess as precisely as possible what the learner achieved accuracy is.

Acknowledgment

The project OPPORTUNITY acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number: 225938.

References

[1] L. Bao and S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, A. Ferscha and F. Mattern, Eds. Springer Berlin / Heidelberg, 2004, vol. 3001, pp. 1–17.

[2] D. Roggen, K. Förster, A. Calatroni, T. Holleczek, Y. Fang, G. Troester, P. Lukowicz, G. Pirkl, D. Bannach, K. Kunze, A. Ferscha, C. Holzmann, A. Riener, R. Chavarriaga, and J. del R. Millán, "Opportunity: Towards opportunistic activity and context recognition systems," in *Proceedings of the 3rd IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC 2009)*. Kos, Greece: IEEE CS Press, June 2009.

[3] M. Kurz and A. Ferscha, "Sensor abstractions for opportunistic activity and context recognition systems," in *5th European Conference on Smart Sensing and Context (EuroSSC 2010), November 14-16, Passau Germany*. Berlin-Heidelberg: Springer LNCS, November 2010, pp. 135–149.

[4] M. Kurz, G. Hölzl, A. Ferscha, A. Calatroni, D. Roggen, G. Tröster, H. Sagha, R. Chavarriaga, J. del R. Millán, D. Bannach, K. Kunze, and P. Lukowicz, "The opportunity framework and data processing ecosystem for opportunistic activity and context recognition," *International Journal of Sensors, Wireless Communications and Control, Special Issue on Autonomic and Opportunistic Communications*, vol. 1, June 2011.

[5] M. Kurz, G. Hölzl, A. Ferscha, H. Sagha, J. del R. Millán, and R. Chavarriaga, "Dynamic quantification of activity recognition capabilities in opportunistic systems," in *Fourth Conference on Context Awareness for Proactive Systems: CAPS2011, 15-16 May 2011, Budapest, Hungary*, May 2011.

[6] A. Calatroni, D. Roggen, and G. Tröster, "Automatic transfer of activity recognition capabilities between body-worn motion sensors: Training newcomers to recognize locomotion," in *Eighth International Conference on Networked Sensing Systems (INSS'11)*, Penghu, Taiwan, Jun. 2011.

[7] J. A. Ward, P. Lukowicz, G. Tröster, and T. E. Starner, "Activity recognition of assembly tasks using body-worn microphones and accelerometers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1553–1567, 2006.

[8] A. K. Dey and G. D. Abowd, "The context toolkit: Aiding the development of context-enabled applications." ACM Press, 1999, pp. 434–441.

[9] T. van Kasteren, G. Englebienne, and B. Kröse, "Transferring knowledge of activity recognition across sensor networks," in *Pervasive Computing*, ser. Lecture Notes in Computer Science, P. Floréen, A. Krüger, and M. Spasojevic, Eds. Springer Berlin / Heidelberg, 2010, vol. 6030, pp. 283–300.

[10] W. Dai, Y. Chen, G.-R. Xue, Q. Yang, and Y. Yu, "Translated learning: Transfer learning across translated learning: Transfer learning across different feature spaces," in *Conference on Neural Information Processing Systems 2008 (NIPS'08)*, December 2008, pp. 353–360.

[11] D. Roggen, A. Calatroni, M. Rossi, T. Holleczek, K. Förster, G. Tröster, P. Lukowicz, D. Bannach, G. Pirkl, A. Ferscha, J. Doppler, C. Holzmann, M. Kurz, G. Holl, R. Chavarriaga, M. Creatura, and J. del R. Millán, "Collecting complex activity data sets in highly rich networked sensor environments," in *Proceedings of the Seventh International Conference on Networked Sensing Systems (INSS), Kassel, Germany*. IEEE Computer Society Press, June 2010.

[12] Y.-l. Chou, *Statistical analysis, with business and economic applications*. Holt, Rinehart and Winston (New York), 1975, vol. 2nd Edition, no. 0030894220.

 ISBN: 978-1-61208-156-4

Self-Adaptive Agents for Debugging Multi-Agent Simulations

Franziska Klügl
Modeling and Simulation Research Center
Örebro University
Örebro, Sweden
Email: franziska.klugl@oru.se

Carole Bernon
Institut de Recherche en Informatique de Toulouse
Paul Sabatier University (Toulouse III)
Toulouse, France
Email: carole.bernon@irit.fr

***Abstract*—In this contribution, we propose an adaptation-driven methodology for the technical design and implementation of multi-agent simulations that is inspired by the concept of "living design". The simulated agents are capable of evaluating their behavior and self-adapt for improving the overall model. For this aim, the modeler describes critical, non valid situations in the life of an agent, or the complete agent system, and explicitly specifies repair knowledge for these situations.**

***Keywords*-multi-agent systems, simulation, methodology.**

I. Introduction

The development and usage of appropriate methodologies for developing multi-agent simulations has become a focus of scientific attention as it became obvious that traditional approaches are not sufficient to handle the complexity of design, implementation, testing or validation of multi-agent simulations as they do not account for generative nature of these form of simulation models. Our paper contributes to this research on techniques supporting multi-agent simulation development. We propose a procedure where agents themselves improves the quality of the technical design and implementation of an agent-based simulation model. The part of the process that we address is often also referred to as debugging in a quite wide sense.

The development of such a high quality model is challenging: first of all, agent-based models are *generative* [1]. That means the overall dynamics is generated from micro-level agent behaviors and interactions. There is no explicit relation between what is defined in the model and what is the produced overall behavior, but this relation is just established by simulation. Thus, there is a profound uncertainty on assumptions mainly about agent-level structures and behavior model but also on the constituents of the environmental model. The appropriateness of assumptions can only be tested and determined using a fully calibrated, runnable simulation. On a more technical level, other challenges have to be taken up such as the mere size of the simulation in terms of number of agents or their level of heterogeneity. The brittleness of a model coming from non-linear interaction outcomes and from the usage of so-called "knife-edge thresholds" [2] in agent models may also aggravate the general problem. From a practical point of view, the size of a model with many parallel acting and interacting agents, with different categories of agents in different local contexts causes problems for implementation: it is problematic to oversee the dynamics, to identify problems and locate bugs in the implemented model.

In this contribution, we propose an approach for design and implementation of a multi-agent simulation model that is inspired by the "Living Design" concept [3]: in addition to the behavior that shall be actually simulated, the agents in the simulation are equipped with an additional meta-level module for monitoring their performance, identifying problems and repairing them. The necessary knowledge is provided by the modeler either explicitly, for example, in terms of constraints, or implicitly by interacting with the simulated agent. Our suggestion is appropriate for those agent-based simulations that possess an original system and are developed with a clear question; in terms of [4] a case study rather than a model abstraction.

After a short introduction to the background of this work, the concept of self-debugging agents is introduced and discussed. After comparing our approach to related works, a conclusion and some outlook to future work are given.

II. Agent Adaptation instead of Manual Repair

Consider the following situation: a modeler has to develop a multi-agent evacuation simulation where pedestrian agents have to exit at best a train or a building. The initial test runs produce a number of situations where agents are blocking each others at exits or bottlenecks, agents are getting stuck with obstacles. Clearly, the behavior model of the agents is not valid, that means it is neither optimally calibrated nor complete or correct. In a painful trial and error procedure, the modeler now might adjust minimal acceptable distances, add behavior elements for explicitly giving way to others or for pushing past the other agents. Our idea is to avoid this iterative manual improvement by the modeler by giving the pedestrian agents the capability to recognize such critical situations and modify their behavior and the elements of the simulated environment to avoid them in the future.

In the Agent-Oriented Software Engineering (AOSE) area, *Adelfe* [5], related to the concept of Adaptive Multi-Agent Systems (AMAS) [6], is one of the rare methodologies that explicitly suggests using self-designing agents to build

 ISBN: 978-1-61208-156-4

a multi-agent system. Based on meta-rules describing the prerequisites for "useful" interacting agents, the AMAS theory identifies 7 categories of "Non-Cooperative Situations" (NCS) in that these prerequisites are not fulfilled. NCS have to be anticipated, avoided or eventually repaired by every agent. The processing of NCS is separated from the actually intended behavior of an agent. It possesses a "nominal" behavior for performing the usual behavior and an additional meta-level "adaptive" behavior to discover and eliminate the NCS it encounters. Thus, the agent learns to adapt to its environment when the nominal behavior is not sufficient.

Transferred to the problem of technical design and implementation in multi-agent simulation, NCS correspond to situations that do not occur in the original system. We call these Non-Valid Situations (NVS) for denoting the particular relation to producing valid simulations. They are not restricted to blocking situations as sketched in the beginning, but may also characterize discrepancies between simulated and real data on a macro-level, the missing production of organizational structures, etc. Situations that might be judged as non-cooperative for an artificial multiagent system may actually occur in the real world and thus must occur in the corresponding context in the simulation. Looking through a developers eye, there is no basic conceptual difference between NCS and NVS; Nevertheless, we think that "NVS" is more appropriate in the simulation context focusing on the model development level.

III. Adaptive Agents for Model Debugging

A. General Concept

The general concept of a self-debugging agent is based on the idea that an adaptation component can be added to the standard agent model for identifying and repairing NVS. The respective phases (illustrated in fig. 1) are the following:

1) The starting point is a runnable initial prototype for the full agent-based simulation model.
2) The modeler develops the adaptive module for identifying NVS and how the simulated agents deal with these critical situations.
3) The original nominal agent behavior is connected to the adaptive behavior and both are running concurrently while the simulation is executed and repeated.
4) The nominal model and the adaptive behavior are separated again when all identified NVS are solved. Parts of the adaptive elements may remain in the agent program.
5) The validity of the final model is tested thoroughly again. If previously hidden or new NVS are discovered, an adaptive behavior has to be added and the adaptation starts again, otherwise the model is ready for final documentation and deployment.

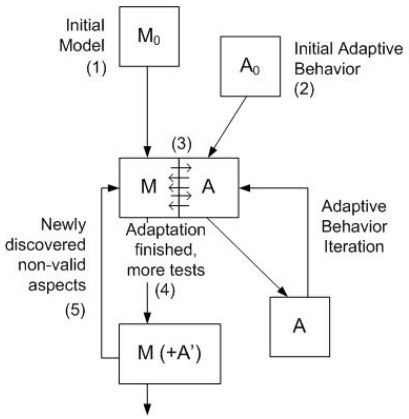


Figure 1: Adaptation-driven design process starting from an initial model M_0 and a corresponding adaptive repair component A_0. After iteratively testing and adapting, an improved model is produced that may contain some built-in adaptive elements.

B. Agent Architecture

The particular agent architecture for self-debugging agents consists of two parts: the original nominal behavior and the adaptation module. All agents, a subpopulation or even just one agent, may be equipped with an adaptation component. In principle, also entities without agent properties – such as resources, obstacles or the modeled environment – may be equipped with an adaptation component that modifies their parameters. This enables handling adaptations that are not isolated to an entity but need to affect more than one entity in a coordinated fashion. The overall architecture is sketched in Figure 2.

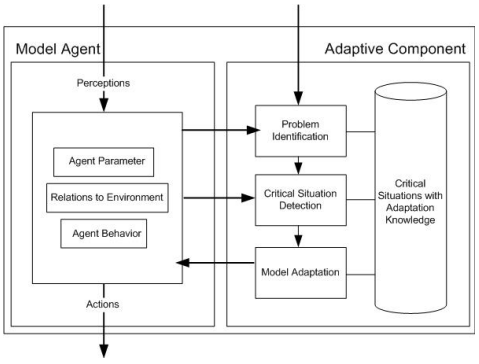


Figure 2: Architecture of a self-debugging agent with both nominal (left side) and adaptive behavior (right side).

The adaptation module has its own perception that may include the agent's status and perceptions, but also additional perceptions from the simulated environment or even connections to external data sources. The actions of the adaptation module may directly effect only the agent to which it is assigned.

 ISBN: 978-1-61208-156-4

C. Self-Debugging Phases

In the following, the three major (iterated) tasks that a self-debugging agent has to execute are discussed. First, the adaptation module has to notice that something in the simulation is going wrong. The identified problem has to be reduced to one or more NVS that are basically capturing assumptions for possible causes for the problems. Instructions for repairing the model in reaction to the NVS are then associated with the NVS. In the third phase, the instructions are executed and the model is adapted. This cycle has to be repeated again and again until no problem is observable any more.

A problem might be the availability of appropriate meta-knowledge to run this cycle. This problem is depending on the particular model that is to be debugged. With an appropriate – model depending – interface between adaptive module and human system expert, human intelligence and implicit knowledge can be used for filling the gaps in explicitly formulated knowledge.

1) Problem Identification: The identification of a problem is directly connected to criteria for determining the validity of a simulation model. As we are addressing multi-agent simulation, these criteria can be found on different levels of observation. If a criteria for model validity does not hold, a problem can be detected. Criteria may be both quantitative or qualitative.

1) Most frequently, values, trends and dynamics of global output variables are used for validation: macro-level descriptors are compared to corresponding values obtainable from the simulation. Invariants and similar conditions involving one or more output values may be formulated as a criterion of model validity that must be satisfied. An adaptation module that should be capable of identifying a problem on the macro-level needs a global perspective.
2) Output values may serve as a basis for comparison also on a meso (group) or micro (individual agent) level. The latter may be important if prominent, unique agents are part of the simulation.
3) Macro-level structures, that shall emerge from the interaction of the whole collective of agents, may be characterized. Their failure to express emergent behavior or the possible inadequateness (shape, time, etc.) of this latter may lead to identification of a problem.
4) Interaction outcomes can be characterized and may form a source for criteria determining whether a simulation is not running smoothly. Agents block each other where they should not, do not interact when they should or interact with the wrong partners.
5) A similar source of criteria for validity are individual behavior paths. Agents decide for other than the rational option or stop when they should not.

Clearly, these criteria are not independent from each other as macro-level outcomes are generated from the micro-level. Nevertheless, it is useful to define them on different levels of abstraction.

2) NVS Identification: Having perceived that there is a problem, a reason for the problem has to be found for being able to modify the agent behavior in a way that the problem is avoided. As in multi-agent simulations all dynamics are generated by the agents behavior and interactions, the reason for a problem can be found solely on the micro-level. Thus, the original seven NVS of AMAS can also be identified in the context of multi-agent simulations:

1) INCOMPREHENSION. An agent receives an information but has no interpretation for this information. For example, an agent bumps into something, is not able to recognize it as an obstacle and cannot avoid it, getting blocked by it.
2) AMBIGUITY. The perception can be interpreted in more than one way. The behavior model is not refined enough with respect to the environmental complexity or the messages sent by other agents. For example, an agent which is roaming in an environment is not able to distinguish an obstacle from a source of energy, it will therefore be unable to know how it has to behave (avoiding the obstacle or recharging its batteries).
3) INCOMPETENCE. The agent possesses no rule for dealing with a particular internal status. That means the state variables of an agent possess values that are beyond the values that can be dealt with by its reasoning. For example, an agent which wants to recharge and finds a source of energy, finds a source which has a failure, it is therefore incompetent to recharge.
4) UNPRODUCTIVENESS. Given the current information situation, the reasoning is not capable of producing an output. For example, an agent receives a message that is not meant for it, it is not able to do something by using this message.
5) CONCURRENCE. When an agent attempts to execute the selected action, another agent executes the same action resulting in redundancy when this action has just to be done once. If there are these inefficiencies in the original system, they have to be reproduced in the simulation model for developing a realistic reproduction of the original system. Nevertheless there are cases when this redundancy may point to a problem of inappropriate action selection. For example, two agents roaming in an environment want to move in the same place at the same simulated time whereas there is a clear coordination in the real world.
6) CONFLICT. Two or more agents possess a conflict over resources. That means, if one agent executes its actions, the others cannot reach their goals. As with CONCURRENCE, this situation may actually occur,

 ISBN: 978-1-61208-156-4

but real-world agents may have invented mechanisms to cope or solve the conflict. For example, when two simulated pedestrians want to concurrently pass through a narrow door, they are not blocked for a longer time, but one of the agents will proceed while the more polite one will give way. Thus, the task of the adaptive component may not be just to modify the simulated agent for avoiding the critical situation from the beginning, but to add appropriate, timely rules about how agents can decide in the case of conflict.

7) USELESSNESS. The action of an agent has no effect. In the simulation case, it might happen that the action has no impact neither on the environment nor on the internal status of the agent. For example, a car agent that travels between two locations in a town environment is blocked because of a traffic jam at a unrealistic position. Being obliged to stay still is useless for the agents since this situation does prevent it (and may be others that are blocked behind it) from reaching its goal.

3) Adaptation: For every identified problem there might be a number of critical situation descriptions for explaining the identified problem. With a critical situation serving as explanation, there should be an assigned adaptation or "repair" plan – a sequence or skeletal plan – stating what can and shall be modified for tackling the critical situation. The selection of the modifying actions is not necessarily deterministic. Optimality and convergence can only be determined for a particular model.

One can identify the following action repertoire usable in such a plan:

1) *Move*(<*destination*>) modifies the agents embedding into its environment – move changes its local context by moving the agent to another location. The movement has to be seen figuratively: movement not just in metric space, but also within networks and organization, adapting addressees of information, communication partners, etc.
2) *AdaptParameter*(*Parameter,* [<*direction*>]) modifies an individual parameter of the modeled agents. For example, *AdaptParameter*(*DesiredSpeed,* $[Increase]$) increases the parameter that determines the desired speed of an agent. This might lead to more heterogeneity in the agent population as every agent modifies its parameters individually. However, if the parameter of all agents have to be adapted, this individual-based adjustment may not be efficient.
3) *InsertRule*(<*condition*>→<*action*>) means structurally modifying the agent behavior by adding rules For discovering the appropriate model modifications, agents may use learning techniques.
4) *Generate*(<*agentType*> $[at$ <*time*>*]*) generates new agents to the current situation at a given time. This time may be "now" or any point during the simulation or the start situation. For reproducibility, this generation must be integrated into the model description – therefore changing the model input adding a new external event from a local perspective. Since the simulated environment is treated as an explicit entity in the simulation model, it may also be equipped with an adaptive component changing the input behavior from a global perspective.
5) *Delegate*(<*situation*>→<*list of agents*>) notifies a set of other agents about the occurrence of a specific critical situation. Basically, the agent has detected that there is a particular critical situation that cannot be solved by itself but by other agents involved in the situation.
6) *Delegate*(*ResponsibleModeler*). The adaptive component of the model entity has detected that there is a specific situation, but has no information or plan for coping with the situation. Thus, it notifies the modeler, or the domain expert, about the occurrence of the situation and asks for modifications by the human involved. Such a "repair" plan element has similarity to some break points in traditional programming languages.

These actions for modifying the simulated agents nominal behavior can be configured and used to set up a repair plan. This has to be done for each identified NVS. The identified NVS gives the particular addressee (particular parameter, network position, etc.) of the modifications. However setting up such a repair plan may not be simple. As indicated in the last possible action, a solution might be delegating the problem solution to an involved human expert.

After modification, whether the modified agents are now able to produce the intended overall behavior has to be tested. Depending on the particular simulation endeavor, the simulation run has to be completely restarted or can be resumed from the simulated time where it was stopped due to the identification of the problem. When blocking situations are involved, resuming might be sufficient to see whether the USELESSNESS or CONFLICT situation resolves. On the other side, when population dynamics are involved, restart may not be avoidable.

IV. RELATED WORKS

Due to their main features – distribution, heterogeneity, parallelism, scalability, autonomous behaviors of agents, emergent collective behavior, etc. – multi-agent systems are difficult to test and debug. Agent-oriented methodologies are becoming mature enough to study how phases of testing and validation can be integrated into their process development [7]. However, very few are specifically interested in guiding the development of agent-based simulations and less aim at helping the modeler by providing him with tools to facilitate the implementation of the model, to observe how

 ISBN: 978-1-61208-156-4

it behaves and to improve this behavior. In *Ingenias*, for instance, a graphical modeling language helps modelers to design a model and code generation is then enabled toward some simulation platforms (Mason, Repast) [8]. In *Adelfe*, a fast prototyping ability is offered to see how the agents behave, however this tool is based on state machines and does not offer a visual aid that would enable detection of wrong behaviors by observing agents moving in a simulated environment [5]. Also in [9] agent self-modification using is suggested for adapting to a dynamic environment.

Debugging in multi-agent systems is often provided by observation and visualization tools, see [10] for a suite of visualization tools devoted to debugging. However, displaying a huge quantity of information to the designer, most of the time about interactions between agents, is not always helpful and automatic correlation or/and analysis of these data (e.g., using graph as in [11]) or automatic detection of wrong behaviors is highly desirable. In Prometheus, this help is brought by a tool, which observes conversations between agents, tests if they conform to the specified interaction protocols, and notifies violations to the developer [12]. This kind of debugging can only validate specifications initially made during the design and cannot discover ways of communicating that could enable a better behavior at the micro or macro-levels. Since agent-based systems can be considered from different perspectives, the approach proposed in [13], for notably testing simulations, is to study problems (mainly deadlocks, bottlenecks and performance problems) that can occur at the agent level before considering errors at the interaction level for eventually testing problems at the system one. To avoid a deterministic approach, tests are made in a stochastic way and repeated several times. Here also, testing at an agent level compares its behavior with the way it was specified and designed; may be the result of testing may lead to notify the designer of problems but nothing is done to enable discovering a behavior better adapted to some constraints or environment.

As discussed in the beginning, the situation for multi-agent simulation is difficult as testing and debugging for implementation bugs is only secondary. Due to their generative nature and often restricted data availability, "guessing" about the agent level behavior and interaction is based on experience and creativity. Explicit handling of information on the original system is central. Most suggested tests for simulations are therefore based on humans evaluating model structure, behavior and outcome [14]. Our suggestion is a realistic intermediate step between full human-done modeling, implementing and debugging and approaches such as [15] that try to transfer the agent modeling problem to a learning problem.

V. Discussion

There are aspects in the proposed approach to debugging in multi-agent simulation that need to be discussed. The most important issues are the availability of explicit meta-knowledge, dependencies between adaptations, the convergence of adaptation and the simplicity and understandability of the outcome.

The approach described so far can only work when criteria for validity can be formulated and tested automatically by the agents, and changes be given on how to react on the perceived deficiencies. The former is especially depending on the particular model. For many multi-agent simulations only statistical data are available on the macro level, population numbers, average turnovers, road link load over a certain time, etc. However, for the described approach, the validity criteria have to be broken down to individual values, behavior paths or the outcomes of interactions. Whether such information is formulate-able is depending on the particular model. However, often a modeler, domain expert or stakeholder can, based on their experience with the original system, identify situations and features that do not "look" realistic although they might not be able to explicitly describe what are the reasons for this invalid observations. When having identified that there is a problem, they can often tell how the situation should "look" like or what should happen. Thus, based on an appropriate user interface, human intelligence should be used when automatic detection and repair is not possible. Actually, this would be a fall back to the previous way of debugging, it is now embedded into systematics.

There is a second critical aspect: one can find application scenarios where model modifications cannot be done for agents independently from modifications in other agents or entities. For example, the metabolism of an agent has a connection to available resources in the environment when intending to produce some sustainable agent population growth. Interdependent modifications in different model elements cannot be formulated based on adaptive components added to individual agents. Higher level entities can be created that do not have a role, or just may have a passive role in the original model. These entities or the explicitly modeled environment can be augmented with the meta-level module for testing and adaptation.

A priori, we cannot guarantee that the proposed approach has any noticeable impact on model quality properties such as simplicity or understandability. These are highly depending on the way the modeler formulated the initial nominal and the adaptive behaviors. However, the distinction between nominal and adaptive behavior and their explicit handling may support understandability, as the modeler is forced to explicitly treat and add meta-level knowledge about the model. We would also expect that the explicitness of such information is also of great relevance for the reusability and maintainability of the model.

 ISBN: 978-1-61208-156-4

VI. Conclusion

We proposed a concept and architecture for self-debugging agents for multi-agent simulation. A modeler can start with a model prototype, augment it with additional explicit knowledge on how the outcome looks like. Then, systematically non-valid situations are identified and repaired. This forms an iterative, yet consequent debugging and modification process towards a sufficiently valid representation of the original system. Our framework relies on the ability of a human modeler to identify and formulate critical situations. Yet, yhe modeler does not need to provide direct solutions about how to adapt the agents' behavior for coping with or avoiding such situations, but just needs to indicate what the agents might have to change for improving future simulation runs. Thus, the process is different from trial and error debugging but supports systematic design. From the beginning, the modeler has to think about potential critical situations. He explicitly classifies what may be adapted and what is sacrosanct. This is not only a deliberate treatment of potential non valid situations but also a clear and concise elaboration of what are the parts of a model that are fixed (e.g., because the data are clear or the underlying theory does not allow adapting these model elements) or which ones can be varied to what extent.

Clearly there are some weaknesses and potential perils that have to be analyzed and tackled in future research: in the current approach, there are only local adaptation actions suggested, initiated by agents with local view on NVS. A solution might be introducing adaptive agents on higher aggregation levels. The most critical issue is whether the modeler actually can provide sufficient meta-level information for specifying the adaptive part of the agent behavior, the NVS including the repair plans. If he just can specify the NVS but has to delegate all repair actions to a manual modification, the overall process is reduced to defining constraints and invariants and automatically testing them, notifying the modeler in cases that they are violated. This is nevertheless a valuable modeling support. Our future research will therefore be directed to the application of the self-modeling agents in a variety of application domains gathering experience about forms of accessible and explainable meta-knowledge that can be used for systematically defining critical situations and repair plans. Also, we will test data mining methods for supporting the elicitation of model meta-knowledge that can be used for guiding the modifications towards better multi-agent models.

References

[1] J. M. Epstein, "Agent-based computational models and generative social science," *Complexity*, vol. 4, no. 5, pp. 41–60, 1999.

[2] L. R. Izquierdo and J. G. Polhill, "Is your model susceptible to floating-point errors?" *Journal of Artificial Societies and Social Simulation*, vol. 9, no. 4, p. 4, 2006.

[3] J.-P. Georgé, G. Picard, M.-P. Gleizes, and P. Glize, "Living design for open computational systems," in *Int. Workshop on Theory And Practice of Open Computational Systems (TAPOCS at WETICE 2003), Linz, June, 2003*. IEEE Computer Society, 2003, pp. 389–394.

[4] R. Boero and F. Squazzoni, "Does empirical embeddedness matter? methodological issues on agent-based models for analytical social science," *Journal of Artificial Societies and Social Simulation*, vol. 8, no. 4, p. 6, 2005.

[5] C. Bernon, V. Camps, M.-P. Gleizes, and G. Picard, "Engineering adaptive multi-agent systems: the Adelfe methodology," in *Agent-Oriented Methodologies*, B. Henderson-Sellers and P. Giorgini, Eds. Idea Group Pub, June 2005, pp. 172–202.

[6] M.-P. Gleizes, V. Camps, and P. Glize, "A theory of emergent computation based on cooperative self-organization for adaptive artificial systems," in *4th European Congress of Systems Science*, L. Ferrer Figueras, Ed. Spanish Society of System Science, 1999.

[7] B. Henderson-Sellers and P. Giorgini, *Agent-Oriented Methodologies*. Idea Group Pub, June 2005.

[8] C. Sansores and J. Pavón, "Agent-Based Modeling of Social Complex Systems ," in *Current Topics in Artificial Intelligence (CAEPIA 2005)*, ser. LNAI, R. Marín, E. Onaindía, A. Bugarín, and J. Santos, Eds., vol. 4177. Springer, 2006, pp. 99–102.

[9] F. Brazier and N. Wijngaards, "Designing self-modifying agents," in *Proc. of the Creative Design Workshop, Dec. 2001*, 2001.

[10] D. T. Ndumu, H. S. Nwana, L. C. Lee, and J. C. Collis, "Visualising and debugging distributed multi-agent systems," in *Proc. of the 3rd Conf on Autonomous Agents*. New York, NY, USA: ACM, 1999, pp. 326–333.

[11] E. Serrano, J. J. Gómez-Sanz, J. A. Botía, and J. Pavón, "Intelligent data analysis applied to debug complex software systems," *Neurocomputing*, vol. 72, no. 13-15, pp. 2785–2795, 2009.

[12] L. Padgham, M. Winikoff, and D. Poutakidis, "Adding debugging support to the prometheus methodology," *Engineering Applications of Artificial Intelligence*, vol. 18, pp. 173–190, 2005.

[13] T. Salamon, "A three-layer approach to testing of multi-agent systems," in *Information Systems Development*, G. A. Papadopoulos, W. Wojtkowski, G. Wojtkowski, S. Wrycza, and J. Zupancic, Eds. Springer US, 2010, pp. 393–401.

[14] O. Balci, "Validation, verification and testing techniques troughout the life cycle of a simulation study," *Annals of Operations Research*, vol. 53, pp. 121–173, 1994.

[15] R. Junges and F. Klügl, "Evaluation of techniques for a learning-driven modeling methodology in multiagent simulation," in *Proc. of the 6th Int. Conf. MATES, Leipzig*, 2010.

 ISBN: 978-1-61208-156-4

Mathematical Model for the Optimal Utilization Percentile in M/M/1 Systems:

A Contribution about Knees in Performance Curves

Francisco A. González-Horta, Rogerio A. Enríquez-Caldera, Juan M. Ramírez-Cortés, Jorge Martínez-Carballido
INAOE, Department of Electronics
Luis Enrique Erro #1, Tonantzintla, 72840, Puebla, México. {fglez, rogerio, jmram, jmc}@inaoep.mx

Eldamira Buenfil-Alpuche
Master in Computer Science
Professional Associated to INAOE
eldamira@gmail.com

***Abstract*— Performance curves of queuing systems can be analyzed by separating them into three regions: the flat region, the knee region, and the exponential region. Practical considerations, usually locate the knee region between 70-90% of the theoretical maximum utilization. However, there is not a clear agreement about where the boundaries between regions are, and where exactly the utilization knee is located. An open debate about knees in performance curves was undertaken at least 20 years ago. This historical debate is mainly divided between those who claim that a knee in the curve is not a well-defined term in mathematics, or it is a subjective and not really meaningful concept, and those who define knees mathematically and consider their relevance and application. In this paper, we present a mathematical model and analysis for identifying the three mentioned regions on performance curves for M/M/1 systems; specifically, we found the knees, or optimal utilization percentiles, at the vertices of the hyperbolas that relate response time as a function of utilization. Using these results, we argue that an adaptive and optimal queuing system could be deployed by keeping load and throughput within the knee region.**

Keywords- adaptive queuing system; performance optimization; knees in performance curves; optimal utilization region; optimal throughput region

I. INTRODUCTION

According to Bose [1] and Kleinrock [2], queuing is a basic phenomenon that arises whenever a shared resource (server) of finite capacity is accessed for service by a large number of jobs or customers. Queues or waiting lines are frequent in many systems and daily life situations; e.g., when you wait for a free ATM to take money out of your bank account, or when a computer inputs data packets into the network, and after a time delay, they are delivered to the destination computer.

Nobody likes to wait too long for a service on a shared resource, thus, one major goal for the performance of a queuing system is to reduce, as much as possible, the system *response time* (R) or delay. However, making a very fast service may have a very high cost. The response time may be reduced if the system capacity is increased, but, if we extend the capacity more than necessary, then the costs for system maintenance and construction will rise, and surely, many resources will be wasted unnecessarily. Therefore, in order to reduce those costs, the system should operate with the greater load or throughput as possible, i.e., allowing to attend as many jobs or customers as possible per unit of time.

One measure of load is *utilization* (U), which is the resource usage divided by resource capacity for a given time interval. Thus, increasing the utilization rate of a queuing system is another relevant goal for raising its performance. However, as utilization for a resource goes up, so does the response time, meanwhile in the opposite way, as the response time goes down, so does the utilization rate. This means that there is a conflict between reducing the response time and increasing the system utilization, both goals cannot be optimized simultaneously, unless we find a compromised or balanced solution. In optimization theory, a Pareto improvement [3] can be made by improving one goal as long as the change that made that goal better off does not make the other goal worse off. When no further Pareto improvements can be made, then the solution is called Pareto optimal or non-dominated solutions. The utilization percentile at which this optimal balance occurs is called the *knee*. This is the point at which load is maximized with minimal negative impact to response times.

A recent article by Cary Millsap [4], about performance for computer software, brought back a discussion on an old debate that has been rounding out the queuing literature for more than 20 years. In 1988, Stephen Samson [5] argued that, at least for M/M/1 queuing systems (i.e., single-server queues where both inter-arrival times and service times follow the exponential distribution), no "knee" appears in their performance curves. Moreover, Samson wrote: "In most cases there is not a knee, no matter how much we wish to find one." Since that moment, a historical debate was initiated between those who support the Samson's claim about knees, e.g., [6][7], and those who argue the existence and relevance of the "knee in the curve" [4]. In this paper, we provide a mathematical approach based on differential calculus to find a "knee" or optimal balance point between conflicting goals involving minimizing response times and maximizing utilization rates for M/M/1 systems. In difference with Millsap [4], who located the knee utilization for a single-server system at U = 50% independently of any other performance parameter, we claim that the knee location is dependent on the average service time for each arrival (S). This knee occurs where the hyperbola, relating U

 ISBN: 978-1-61208-156-4

and R, makes its sharpest turn, corresponding to its vertex located at the point $(U, R) = (1 - \sqrt{S}, \sqrt{S})$.

This paper is organized as follows. Section II presents an overview of proposals arguing in favor or against the existence and location of knees in performance curves. Section III starts by presenting a synopsis of the M/M/1 queue model and its nomenclature, then, it shows the mathematical model and analysis for the optimal utilization and the optimal throughput in M/M/1 performance graphs. Thereafter, we propose a region of optimality based on the hyperbola latus rectum. Section IV discusses the relevance of the knee concept and its application to adaptive and optimal communication networks. Finally, Section V concludes the paper with a summary of contributions and future work.

II. OPEN DEBATE ABOUT KNEES

A. Proposals Against the Existence of Knees

Neil Gunther in [6] makes a rigorous but unconventional study about knees. He, in fact, analyzed several of the concepts we use in this paper. For instance, he described the hyperbola vertex as an optimum, and he used the endpoints of the latus rectum to find alternative optimum points. Surprisingly, he arrived to the following conclusion: there is no "knee" on the response time curve, even in the case of M/M/1 systems; the same conclusion that Samson arrived in 1998.

A detailed analysis of Gunther's argumentation revealed that he arrived to such conclusion because he analyzed only a normalized response time function (R/S). The R/S function equals to 1/(1–U) which corresponds to the R(U) function with S = 1. As we show in this paper, the curve for S = 1 is one of the most inefficient performance curves, because such curve is for an unconventional large number of service times. This is the reason why Gunther, declined his interest in considering the vertex of a hyperbola as the knee in the curve.

Ley [7] reviewed ten different definitions about the knee concept, but, he concluded that there is no a clear definition of what constitutes the knee in the curve, and that all the definitions he collected do not agree with the traditional 70% utilization level. As we prove in this paper, the traditional 70-90% for the optimal utilization range is a myth, because it depends on the service capacity, which makes such a value not a universal constant.

B. Proposals in Favor of the Existence of Knees

Millsap [4] argues in favor of the existence of knees in performance curves. His paper in fact is quite motivating and provides many useful insights into the fundamentals of performance and further details about this historic debate. He published, in that paper, a table of knee values expressed in utilization percentiles for different number of servers in M/M/m systems. Particularly, for M/M/1 systems he claimed the knee value is 50%. He mentioned that the knee values for an arbitrary number of servers are difficult to calculate, but he also said that the only parameter required to compute them was the number of service channels or servers. However, we disagree on that issue because as we show in this paper, the sharpest point in the curve is dependent on the capacity of the system. We deduced that he obtained such knee values by minimizing the function R/U defined for a specific number of servers. Obtaining the turning points for such function, i.e., making d(R/U)/dU = 0, he calculated the U value of 50% independent of S. The problem with dividing the R function by U is that the R/U function is undefined at U = 0, which is inconsistent with the valid value of R at no load.

III. MATHEMATICAL MODEL AND ANALYSIS

A. The M/M/1 Queue Model and Parameters

The simplest queuing system is represented by the Kendall notation as M / M / 1 / ∞ / FCFS. This means that customers arrive according to a Poisson process (first M), they request exponentially distributed service times from the server (second M), the system has only one server, an infinite waiting queue, and customers are served on a First Come First Served (FCFS) basis. For simplicity, this queuing system is sometimes named an M/M/1 system.

As it was neatly described by Chee-Hock in [8], the single-server queue is a place where customers arrive individually to obtain service from a service facility. The service facility contains one server that can serve one customer at a time. If the server is idle, the customer is served immediately. Otherwise, the arriving customer joins a waiting queue. This customer will receive his service later, either when he reaches the head of the waiting queue or according to some *service discipline*. When the server has completed serving a customer, the customer departs. Along this paper, the generic terms 'customers' and 'servers' are in line with queuing literature, but they take various forms in different application domains; e.g., in the case of a data switching network, 'customers' are data packets and 'servers' are the transmission channels.

The M/M/1 system is depicted in Fig. 1. This figure also illustrates some important parameters associated with the queuing model. We describe them briefly.

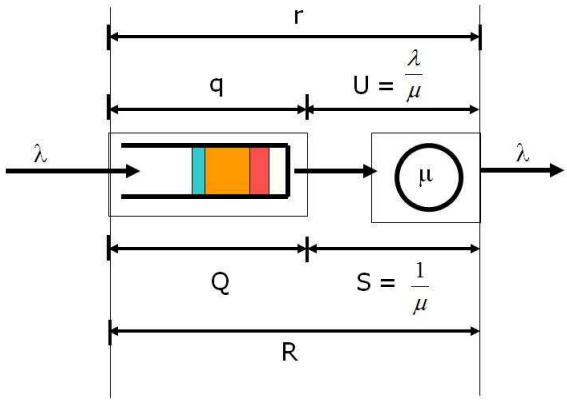


Figure 1. The M/M/1 queue model and parameters at steady state.

- λ, is the average arrival rate or the mean number of customers arriving at the system per unit of time. In

steady state, the rate for arrivals and departures is the same. Moreover, if the waiting line buffer is infinite, then λ also represents the *throughput* of the system, i.e., the mean number of customers that are served in a single unit of time. The domain of this variable is $\lambda \geq 0$.

- μ, is the average service rate or the mean number of customers that are served by the service facility per unit of time. The operating condition $\lambda < \mu$ states the theoretical maximum input rate for the queuing system at $\lambda_{max} = \mu$. If this condition is not achieved (i.e., if $\lambda > \mu$), the number of customers in the waiting line will grow without limit collapsing the system.
- S, is the average service time per customer, it is defined as the reciprocal of μ; i.e., the time interval between the dispatching of a customer to the server and the departure of that customer from the server. The service time cannot be avoided in real scenarios, thus, $S > 0$.
- U is the utilization rate or the fraction of time in which the server is busy. It is obtained as the arrival rate divided by the service rate and it can be expressed as $U = \lambda S$. For $0 \leq \lambda < \mu$, the domain for U is $0 \leq U < 1$.
- R is the average time that a customer spends in the whole system, waiting and being served; aka, the mean residence time, response time, or delay. $R = 1/(\mu - \lambda)$ or $R = S/(1 - U)$, and the domain for R is $S \leq R < \infty$.
- r is the expected number of customers resident in the whole system, including the customers being served (if any) and the customers waiting (if any). This parameter is defined by $r = \lambda/(\mu - \lambda)$ or $r = U/(1 - U)$ or $r = R\lambda$. The domain for r is $0 \leq r < \infty$.
- Q is the queuing delay or the mean time that a customer spends in a queue waiting to be serviced. $Q = \lambda/(\mu(\mu - \lambda))$, $Q = R - S$, and the range for Q is $0 \leq Q < \infty$.
- q is the average number of customers waiting in the queue. $q = \lambda Q$, $q = \lambda^2/[\mu(\mu - \lambda)]$, $q = r - U$, with $0 \leq q < \infty$.

In M/M/1 queuing systems, the inter-arrival times and the service times follow the exponential distribution, this means that the arrival and service processes are Poisson (or random). The exponential distribution is the only continuous function that has the *memoryless* (M) property, and thus, it is commonly used to model stochastic processes [11]. Examples of random variables that are well-modeled by the Poisson process are: the number of goals in a soccer match, the number of raindrops falling over an area, the time it takes before your next telephone call, the arrival of customers in a queue, etc.

B. The Optimal Utilization Percentile (The Knee)

The performance curves of a queuing system can be obtained by plotting different performance parameters; in particular, we concentrate on relations between R, U, λ, and S or μ. We consider the following relations:

$$R(U;\ S) = S/(1 - U), for\ S > 0\ and\ 0 \leq U < 1. \quad (1)$$

$$R(\lambda;\ \mu) = 1/(\mu - \lambda), for\ \mu > 0\ and\ 0 \leq \lambda < \mu. \quad (2)$$

We use (1) for analyzing the relation between response time R and utilization or traffic intensity U, and (2) for studying the relation between delay R and throughput λ. Equations (1) and (2) also show how the response time is a function of service capacity, described by the service time S in (1) and the service rate μ in (2). We will sketch the graphs of these equations for different capacity parameters.

Fig. 2 shows a plot for R(U; S) illustrating the behavior of response time as a function of utilization. Here, each curve is plotted for different values of S, S = 2 (black), 1, 1/2, 1/4, 1/8, 1/16 (blue). Notice the aspect ratio in the plot is 1-to-1, to avoid what Gunther [9] calls an "optical illusion" produced by using different aspect ratios in the utilization and response time axes, which might result in a misconception about the utilization knees. Notice also that, as Gunther showed in [6], the graph for the response time function can be depicted for an extended range of utilization values $-\infty < U < \infty$, even if it does not make physical sense; therefore, we demarcate at Fig. 2 the actual service utilization range $0 \leq U < 1$ as the region of meaningful performance metrics in between bold blue lines.

Fig. 2 stresses the hyperbolic characteristic of the response time function by extending the utilization range. The gray dotted line highlights the transversal axis of the hyperbolae and pinpoints its sharpest turns or vertices. Notice how each curve becomes sharper as S disminishes or the system capacity increases; however, when $S > 1$, the vertices jump to the negative utilization region and response times within the performance region of interest (blue lines) fluctuate rapidly with small changes of load. It is important to design queuing systems operating with S values lower than 1, whatever the time unit is chosen, e.g., seconds, minutes, hours, etc.

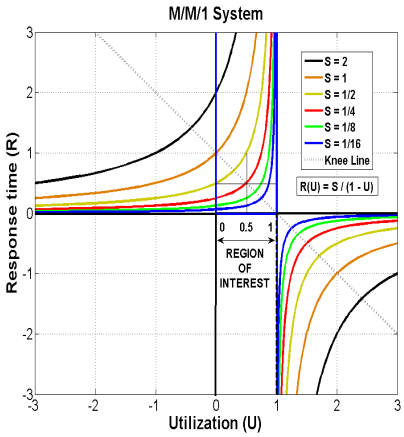


Figure 2. Plot of response time vs utilization for distinct service times.

1) Hyperbola Vertex as an Optimum. Considering the optimality condition that indicates a balance between changes in U and changes in R, the point where a differential increment of U yields the same increment of R, or dR = dU, is the point where the rate of change of R with respect to U (dR/dU) is equal to 1. This condition is reached

exactly at the vertex of a hyperbola. In fact, that is the point that we use to divide the R(U) function initially in three sections: a) the flat section, where the gradient of the curve is dR/dU < 1, indicating a small but constant increment in response time at low load; b) the knee or optimal utilization point, located at the hyperbola vertex, where dR/dU = 1; and c) the exponential section, where dR/dU > 1, indicating that response time rises exponentially at high load.

Our performance goals are maximizing load and minimizing response time. According to Pareto optimality, starting at no load (U = 0), we make improvements or increments in load as long as the resulting increments in response times do not exceed the increments in load (i.e., dR < dU). Thus, the point where optimal balance occurs is the vertex of the hyperbola. Differentiating equation (1) and making *R'(U) = 1*, we obtain the coordinates for the vertices at $(U, R) = (1 - \sqrt{S}, \sqrt{S})$. Notice that the optimal utilization percentile is a function of S, and depending on the service capacity, this optimum occur at U = 0.5, only if S = 1/4; at U > 0.5 (the high-load zone), if S < 1/4; at U < 0.5 (the low-load zone), if S > 1/4; or even at U = 0 if S = 1.

2) Latus Rectum as an Optimal Region. Practical considerations such as response time requirements or buffer sizes, are usually interested in a region of optimal utilization rather than a single optimal point (knee) in the curve. This optimal region is usually located between 70-90% of the theoretical maximum utilization. However, there is no agreement on defining this knee region. We argue that the *latus rectum* of the hyperbola can be used to establish this region of optimality.

Consider Fig. 3. The *Latus Rectum* [10] is the line segment passing through a focus of a hyperbola, which is perpendicular to the transversal axis and has both endpoints (P, Q) on the intersection with the curve (P, Q are indicated just for the curve S = 1).

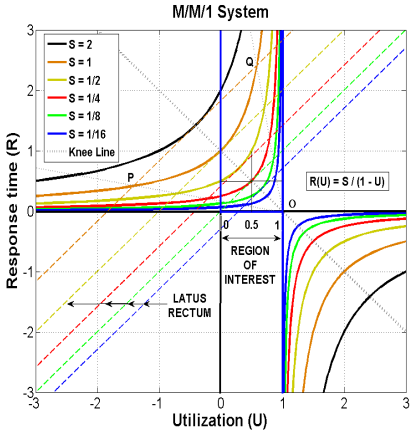


Figure 3. Plot of R(U) for distinct service times (S) with Latus Rectum.

The dashed lines with slope 1 in Fig. 3, represent the graphs of the latus rectum for each sketched hyperbola with S ≤ 1. It can be observed that different scenarios may occur in relation to where the endpoints of the latus rectum are with respect to the region of interest: a) both endpoints are within the bold blue lines, b) one endpoint is in and the other is out, and c) both endpoints are out of the blue lines. The selection of one of these scenarios is, again, dependent on the value of S.

3) Boundaries of the Knee Region

The following analysis is intended to obtain the coordinates for both endpoints of the latus rectum and its length in a general way, such that, it can be obtained for any valid value of S.

We already obtained the coordinates for the vertices, now we obtain the coordinates for the foci intersecting the latus rectum of each hyperbola. The distance from the center O (1,0) to the vertex V $(1 - \sqrt{S}, \sqrt{S})$ of the hyperbola is $OV = \sqrt{2S}$. Using this value, we calculate the distance from center to focus as $OF = 2\sqrt{S}$. Thus, the coordinates for the foci are $F(1 - \sqrt{2S}, \sqrt{2S})$. The equation of the straight line representing the latus rectum is: $R = U - 1 + 2\sqrt{2S}$. Now, intersecting the latus rectum equation with the response time function, we obtain the coordinates for the latus rectum endpoints: $P(1 - \sqrt{2S} - \sqrt{S}, \sqrt{2S} - \sqrt{S})$ and $Q(1 - \sqrt{2S} + \sqrt{S}, \sqrt{2S} + \sqrt{S})$. Thus, the boundaries for the three regions that can be used to study the response time versus utilization performance curves can be stated as: a) *flat region*, the locus of points before P, b) *knee region*, the locus of points between P and Q, and c) *exponential region*, the locus of points after Q. Using the distance between two points we obtain the length of latus rectum as: $2\sqrt{2S}$.

Considering that the region of interest for R(U) is 0 ≤ U < 1 and its knee region is $\left(1 - \sqrt{2S} - \sqrt{S}\right) \leq U \leq (1 - \sqrt{2S} + \sqrt{S})$, we calculate the service capacity condition that makes both endpoints of the knee region reside in the interest region; this condition is $0 < S \leq 3 - 2\sqrt{2}$.

4) Knees in Delay-Throughput Curves

Considering (2), we analyze how the response time R (delay), arrival rate λ (throughput), service capacity μ (bandwidth), and delay knees (hyperbola vertices), are all related to system performance. Fig. 4 shows a plot which we use to illustrate and explain such relation. Here, each hyperbola is plotted for different values of μ, μ = 1 (brown), 2, 4, 8, 16 (blue). Notice again, how the vertices V of the hyperbolas occur precisely when $dR/d\lambda = 1$ and they represent the sharpest points in the curve. Using this condition we locate the vertices of the upper-half hyperbolae at $V(\mu - 1, 1)$. The coordinates for the centers are $O(\mu, 0)$. With the coordinates for O and V, obtain the distance to the vertices as $OV = \sqrt{2}$. Using the distance to the vertex and the definition of hyperbola, we obtain the distance to the focus as $OF = 2$. Knowing the distance to the focus, we obtain the coordinates of the focus at $F(\mu - \sqrt{2}, \sqrt{2})$. Using the coordinates for F and definition of latus rectum, we obtain the equation for latus rectum as: $R = \lambda - \mu + \sqrt{8}$. Intersecting the latus rectum equation with the hyperbola, we obtain the coordinates for the endpoints of the latus rectum at $P: (\mu - \sqrt{2} - 1, \sqrt{2} - 1)$ and $Q: (\mu - \sqrt{2} + 1, \sqrt{2} + 1)$. Thus, the length of latus rectum is $\sqrt{8}$.

 ISBN: 978-1-61208-156-4

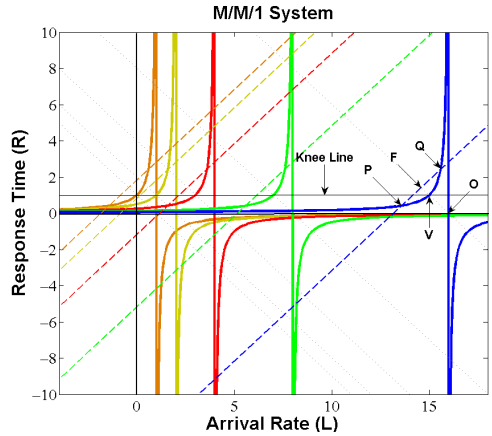


Figure 4. Plot of R(λ) for distinct service rates (μ) with Latus Rectum.

The region of interest is $0 \le \lambda < \mu$. Notice that for low capacity systems, the knee region: $(\mu - \sqrt{2} - 1) \le \lambda \le (\mu - \sqrt{2} + 1)$ may have its lower endpoint outside the region of interest. In order to have both endpoints of the latus rectum within the region of interest, the following condition must be satisfied: $\mu \ge \sqrt{2} + 1$.

System performance is measured by delay R (i.e., the time a customer stays within the system), and by throughput λ (i.e., the number of customers per unit of time that can pass through the system). Throughput is a measure of the system capacity. Delay and throughput are closely related by (2), as throughput approaches 100% of service capacity or bandwidth (μ), delay increases rapidly.

5) Summary of Results

Table I shows a summary of major results obtained from the analysis presented in this paper.

TABLE I. SUMMARY OF MAJOR RESULTS

Useful Results	**Response Times for Load and Throughput**	
	R(U) = S/ (1 – U)	***R(λ) = 1/ (μ – λ)***
Region of interest	$S > 0, 0 \le U < 1$	$\mu > 0, 0 \le \lambda < \mu$
Knee values	$U = 1 - \sqrt{S}$	$\lambda = \mu - 1$
Knee coordinates	$(U, R(U)): (1 - \sqrt{S}, \sqrt{S})$	$(\lambda, R(\lambda)): (\mu - 1, 1)$
Knee regions	$U \ge 1 - \sqrt{2S} - \sqrt{S}$ $U \le 1 - \sqrt{2S} + \sqrt{S}$	$\lambda \ge \mu - \sqrt{2} - 1$ $\lambda \le \mu - \sqrt{2} + 1$
Latus rectum endpoints	$P: (1 - \sqrt{2S} - \sqrt{S}, \sqrt{2S} - \sqrt{S})$ $Q: (1 - \sqrt{2S} + \sqrt{S}, \sqrt{2S} + \sqrt{S})$	$P: (\mu - \sqrt{2} - 1, \sqrt{2} - 1)$ $Q: (\mu - \sqrt{2} + 1, \sqrt{2} + 1)$
Length of latus rectum	$2\sqrt{2S}$	$2\sqrt{2}$
Required service capacity	$0 < S \le 3 - 2\sqrt{2}$	$\mu \ge \sqrt{2} + 1$

These closed-form results show that there is a knee or optimal value for load and throughput in simple M/M/1 queuing systems. Moreover, based on the latus rectum endpoints it is possible to define optimality regions for load and throughput. It can also be observed that both, the knee values and knee regions are dependent on the service capacity of the queuing system. Therefore, it was important for us to determine what would be the required service capacity that will make the knee regions to operate within the regions of interest. The service capacity can be increased by reducing the service time S or by increasing the number of servers. Although we only analyzed the case of increasing the service capacity by reducing S, similar results can be obtained if the case of increasing the number of servers is considered.

IV. RELEVANCE AND APPLICATION OF THE KNEE

A. The Relevance of the Knee Region

Why is the knee value so important? The answer to this question is related to the consequences of having a system operating outside its optimal region. On one hand, if the system operates at the flat region, it is likely that the system capacity is oversized, and therefore, many valuable resources may be wasted. On the other hand, if the system operates at the exponential region, it is likely that the system capacity is undersized, and hence, response times will fluctuate severely even with microscopic changes in load or throughput. The system operation in the exponential region may take the system to instability, oscillating congestion, severe delays, or the worst scenario, to a collapse. Hence, overall on systems with *random arrivals*, it is vital to manage load and throughput so that they do not operate outside the knee region.

At M/M/1 system queues, we do not know *exactly* when the next arrival request or service request is coming; therefore, arrivals have a non-deterministic or random behavior. The M/M/1 model considers an exponential probability distribution for arrivals and service requests. The exponential distribution assumes a higher probability for small inter arrival times. This implies that arrivals will tend to cluster and cause temporary spikes in utilization, as it was mentioned by Millsap [4]. Temporary spikes in utilization beyond the *knee* region may cause serious performance problems or quality of service (QoS) degradation if they exceed a few seconds in duration. This is the reason the knee region is so important on a system with random arrivals.

Once we mentioned the consequences of a system with oversized or undersized capacity, the question is how to determine the adequate capacity that makes a system to operate within its optimal region? *Capacity management* or *capacity planning* is a task intended to answer that question. A first consideration about the service capacity of a queuing system is that it should be calculated so that the optimal region lay down within the region of interest. The last row in Table I shows the conditions for S and μ that make the knee regions for R(U) and R(λ) to be within the region of interest. A second consideration is to estimate the service capacity according to specific service quality requirements; i.e., the expected amount of traffic intensity, throughput, and response times, particularly at peak times. The knee regions

 ISBN: 978-1-61208-156-4

can be computed to meet these QoS requirements. If service capacity cannot be changed dynamically, then the system would operate within its optimal region at peak times, but it would operate outside its optimal region at different load conditions than the peak times. On the contrary, if capacity could be changed dynamically, then the knee regions could change accordingly so that the system operates always within its optimal region. This implies to change *adaptively* the knee regions according to different load conditions (at low load, at peak times, or at excessive high loads). In this way, we believe that and adaptive and optimal queuing system could be deployed by dynamically managing the service capacity in order to keep load and throughput within their knee regions. Therefore, the knowledge about knees is fundamental for capacity management.

B. The Knee Concept Applied to Communication Networks

The M/M/1/∞/FCFS model discussed earlier is simple and useful if we just want to have a first estimation of a system's performance. However, as Chee-Hoc indicates in [8], it becomes a bit unrealistic when it is applied to real-life problems, where they often have a finite waiting queue instead of one that can accommodate an infinite number of customers. A single isolated M/M/1 model may have certain limitations to represent real-life complex queuing systems; however, the networks of queues, whereby the departures of some queues feed into other queues, are a more realistic model for a system with many shared resources interacting with each other. In this way, a model for a virtual circuit in packet switching networks can be designed in terms of a network of tandem M/M/1 queues. Therefore, we can say that M/M/1 queues are the building blocks for all the queuing theory, as we will show in the next discussion.

1) Using the M/M/1 model:

In order to apply the knee concept to a real-life queuing system, we now immerse a bit into the field of communication networks. Two fundamental performance measures of a network are: delay (D) and throughput (T). First notice that we changed the symbols for the equivalent terms we used before as R for delay and λ for throughput in the M/M/1 model. The term delay (or end-to-end delay) specifies how long it takes for a single bit of data to travel across the network cloud from source to destination, as Fig. 5 illustrates. Comer in [12] cleverly describes four types of delays that may occur within the network: propagation delays, switching delays, access delays and queuing delays. All these types of delays contribute to the total delay, which is measured in seconds or fractions of seconds. The other fundamental performance parameter of a network is throughput. Throughput is the rate at which bits of data can pass through the network, and is usually specified in bits per second. The rate at which the hardware can transfer bits is called bandwidth (B). It is impossible for a user to send data faster than the bandwidth; therefore, bandwidth gives an upper bound on throughput or $0 \leq T < B$. Notice that B is equivalent to the μ parameter defined earlier.

In Fig. 5, we also made a distinction between the rate of bits entering the network from source (γ) and the rate of bits leaving the network at destination (T). Notice that for a stable and lossless network, T = γ; however, if T < γ then there will be a steady build-up of bits within the network and it will eventually become unstable. In case of T > γ, new bits might be being created within the network.

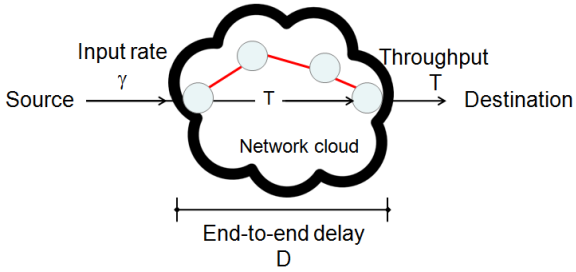


Figure 5. Store and forward data network. Packets traverse from source to destination. Circles are packet switches and red lines are links.

The capacity of a network refers to the maximum number of bits that can simultaneously reside within the network, inclusive of those bits waiting for a shared resource and those that are actually traversing across a link. If D_0 denotes the average delay when the network is idle or when the current utilization (U) is 0, then the network capacity is defined by the Little's Law as D × T (delay-throughput product). According to [8], Little's Law can be applied to all types of queuing systems, including priority queuing and multi-server systems, which is the case for packet-switching networks. Notice that D_0 corresponds to S, U = T/B corresponds to U = λ/μ, and D × T corresponds to r = Rλ in the M/M/1 model. When it is desired to measure the capacity of the underlying hardware, the delay-throughput product becomes the delay-bandwidth product.

Congestion in the network occurs when the offered traffic load from the user to the network produces an excess of bits residing within the network, which exceed the design limit. Data entering a congested network will experience longer delays than data entering an idle network. The relationship between delay and congestion is estimated from the current percentage of the network capacity being used, according to the following expression:

$$D = \frac{D_0}{(1-U)}, for\ D_0 > 0\ and\ 0 \leq U < 1. \qquad (3)$$

As traffic approaches the network capacity (i.e., as U becomes close to 1), the delay approaches infinity. We already plotted this expression in Figure 2 and Figure 3 and obtained some significant insights summarized in Table I.

2) Towards an Adaptive-Optimal Congestion Control:

A simple mechanism for preventing congestion is flow control, which involves regulating the input rate of traffic into the network so that the receiver controls the rate at which it receives data. The main goal of a flow control mechanism is to preserve the levels of throughput and delay within a range of acceptable values. There are basically two types of flow control mechanisms, open-loop and closed-loop. The open-loop flow control is characterized by having

 ISBN: 978-1-61208-156-4

no feedback between the source and destination. This mechanism allocates networks resources for a specific traffic flow with necessary previous reservation. Often this type of mechanism is inefficient and results in over-allocation of resources, it also lacks of any adaptability. On the contrary, the closed-loop flow control is characterized by the ability of the network to report pending network congestion back to the source node. By using this feedback, the source can adapt its transmission rate to a lower rate or a higher rate depending on the network conditions. Real protocols that implement different mechanisms of closed-loop flow control are TCP (Transmission Control Protocol) and ABR (Available Bit Rate). On one hand, TCP adaptively increments or decrements the size of a window in order to speed-up or slow-down the transmission rate. On the other hand, ABR uses congestion information generated in the destination and updated at each packet switch on the path to the source, to reduce or increment the transmission rate accordingly.

Congestion is usually caused by unpredictable events. Although the daily peak hour is semi-predictable, congestion can also be random due to breakdowns, insertions, or changes in the network topology. Therefore, an adaptive mechanism to control the traffic flow is imperative to alleviate congestion problems and preserve the stability of the network.

To know where exactly the borders of the optimal load/throughput region are, as identified in this paper, could be used by the end nodes to regulate and control the stability of the network. If the network capacity could be dynamically managed by the source and destination nodes, then new optimal knee regions would be created, giving the possibility that end stations transmit data flows optimally, i.e. minimizing delays and maximizing throughput [11].

The relevance of the results obtained in this paper is that they can be applied to both, the simplest queuing systems (M/M/1) and the most complex communication network. This is because the M/M/1 model is the building block of any complex queuing system and because Little's Law can be applied neatly to all types of queuing systems.

V. CONCLUSION AND FUTURE WORK

This paper presented a mathematical approach to determine the optimal utilization region and the optimal throughput region for M/M/1 queuing systems. It showed that performance curves for such systems can be analyzed by separating them into three regions: the flat region, the knee region, and the exponential region. The mathematical definition of boundaries between these regions is a problem that has not been properly addressed in the literature. The paper showed that this problem has historical roots and it is still an open debate. The major contribution of this paper is the calculation of knees values and knees regions in performance curves for load R(U) and throughput R(λ). The relevance of knees and their applications was discussed showing the consequences of operating a system outside of its optimality region.

Although the knee model proposed in this paper seems to be mathematically consistent, it is still necessary to create tests, in real applications or simulations, which can help us to validate or invalidate this model. Therefore, our future work will be focused on this task.

ACKNOWLEDGMENT

F.A. González-Horta is a Ph.D. candidate at INAOE. He thanks the financial support received from the National Council of Science and Technology (CONACYT) Mexico, through the doctoral scholarship 58024.

REFERENCES

[1] S. K. Bose, An Introduction to Queueing Systems, Springer, ISBN: 0306467348, 2002.

[2] L. Kleinrock, Computer Science Department, UCLA, Home Page: http://www.lk.cs.ucla.edu/index.html, Retrieved on May 14, 2011.

[3] J. Branke, et al. (Eds.), Multi-objective Optimization: Interactive and Evolutionary Approaches, Springer, Germany, 2008.

[4] C. Millsap, "Thinking Clearly About Performance, Part 2," *Communications of the ACM*, Vol. 53, No. 10, pp. 39-45, October 2010.

[5] S. Samson, "MVS Performance Legends," *Proceedings of Computer Measurement Group Conference*, pp. 148-159, 1988.

[6] N. J. Gunther, "Mind Your Knees and Queues, Responding to Hyperbole with Hyperbolæ," Computer Measurement Group, MeasureIT, published on August 2009. URL: http://www.cmg.org/measureit/issues/mit62/m_62_15.html, retrieved on May 14, 2011.

[7] M. Ley, "Does the Knee in a Queuing Curve Exist or is it just a Myth?" Computer Measurement Group, MeasureIT, published on July 2009. URL: http://www.cmg.org/measureit/issues/mit61/m_61_16.html, retrieved on May 14, 2011.

[8] N. Chee-Hock, and S. Boon-Hee, Queueing Modelling Fundamentals with Applications in Communication Networks, 2nd Edition, John Wiley & Sons, England, 2008.

[9] N. J. Gunther, "Taking the Pith Out of Performance: Watch Your Knees and Queues, Beware of Optical Illusions." Performance Dynamics Company, Blogspot, Published in March 8, 2008. URL: http://perfdynamics.blogspot.com/2008/03/watching-your-knees-and-queues.html , retrieved on May 14, 2011.

[10] Mathwords – Latus Rectum, webmastered by Bruce Simmons; http://www.mathwords.com/l/latus_rectum.htm, retrieved on May 14, 2011.

[11] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi, Queueing Networks and Markov Chains – Modeling and Performance Evaluation with Computer Science Applications, John Wiley & Sons, Inc., ISBN: 0-471-20058-1, USA, 1998.

[12] D. E. Comer, Computer Networks and Internets, 1st. Edition, Prentice Hall, ISBN: 0-13-239070-1, USA, 1997.

 ISBN: 978-1-61208-156-4