ADAPTIVE 2016

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

ISBN: 978-1-61208-463-3

March 20 - 24, 2016

Rome, Italy

ADAPTIVE 2016 Editors

Jaime Lloret Mauri, Polytechnic University of Valencia, Spain

Mario Freire, University of Beira Interior, Portugal

ADAPTIVE 2016

Forward

The Eighth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2016), held between March 20-24, 2016 in Rome, Italy, continued a series of events targeting advanced system and application design paradigms driven by adaptiveness and self-adaptiveness. With the current tendencies in developing and deploying complex systems, and under the continuous changes of system and application requirements, adaptation is a key feature. Speed and scalability of changes require self-adaptation for special cases. How to build systems to be easily adaptive and self-adaptive, what constraints and what mechanisms must be used, and how to evaluate a stable state in such systems are challenging duties. Context-aware and user-aware are major situations where environment and user feedback is considered for further adaptation.

The conference had the following tracks:

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

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

We take here the opportunity to warmly thank all the members of the ADAPTIVE 2016 technical program committee, as well as the numerous reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and effort to contribute to ADAPTIVE 2016. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

Also, this event could not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the ADAPTIVE 2016 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope ADAPTIVE 2016 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of adaptive and self-adaptive systems and applications. We also hope that Rome provided a pleasant

environment during the conference and everyone saved some time for exploring this beautiful city.

ADAPTIVE 2016 Chairs

ADAPTIVE 2016 Advisory Chairs

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

ADAPTIVE 2016 Industry/Research Chairs

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

ADAPTIVE 2016 Publicity Chairs

Kier Dugan, University of Southampton, UK

ADAPTIVE 2016

Committee

ADAPTIVE Advisory Committee

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

ADAPTIVE Industry/Research Chairs

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

ADAPTIVE Publicity Chairs

Kier Dugan, University of Southampton, UK

ADAPTIVE 2016 Technical Program Committee

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

Emiliano Lorini, Institut de Recherche en Informatique de Toulouse (IRIT), France
Sam Malek, George Mason University, USA
Cesar Marin, University of Manchester, UK
Célia Martinie, ICS-IRIT - University Toulouse 3, France
Paulo Martins, University of Trás-os-Montes e Alto Douro (UTAD), Portugal
Mieke Massink, FM&&T Group - CNR, Italy
Olga Melekhova, Université Pierre et Marie Curie - Paris 6, France
Frederic Migeon, IRIT/Toulouse University, France
Vivian Motti, Clemson University, USA
Gero Müehl, University of Rostock, Germany
Christian Müller-Schloer, Leibniz University of Hanover, Germany
Masayuki Murata, Osaka University, Japan
Mats Neovius, Åbo Akademi University, Finland
Filippo Neri, University of Naples "Federico II", Italy
Dirk Niebuhr, Clausthal University of Technology, Germany
Andrea Omicini, Università di Bologna, Italy
Flavio Oquendo, European University of Brittany/IRISA-UBS, France
Mathias Pacher, Leibniz Universität Hannover, Germany
Alexandros Paramythis, Contexity AG, Switzerland
Xingguang Peng, Northwestern Polygtechnical University, China
Georg Püschel, Technische Universität Dresden, Germany
Raja Humza Qadir, dSPACE GmbH, Paderborn, Germany
Claudia Raibulet, University of Milano-Bicocca, Italy
Mahesh (Michael) S. Raisinghani, TWU School of Management, USA
Andreas Rausch, Technische Universität Clausthal, Germany
Wolfgang Reif, University of Augsburg, Germany
Brian M. Sadler, Army Research Laboratory, USA
Yacine Sam, Université François Rabelais Tours, France
José Santos Reyes, Universidad de A Coruña, Spain
Jagannathan (Jag) Sarangapani, Missouri University of Science and Technology, USA
Ichiro Satoh, National Institute of Informatics, Japan
Dominic Seiffert, University of Mannheim, Germany
Huseyin Seker, The University of Northumbria at Newcastle, UK
Sebastian Senge, TU Dortmund, Germany
Estefanía Serral, Vienna University of Technology, Austria
Marjan Sirjani, Reykjavik University, Iceland
Vasco N. G. J. Soares, Instituto de Telecomunicações / Polytechnic Institute of Castelo Branco, Portugal
Christoph Sondermann-Wölke, Universität Paderborn, Germany
Panagiotis Spapis, National and Kapodistrian University of Athens, Greece
Stephan Stilkerich, Airbus Group Innovations, Germany
Martin Swientek, Capgemini Deutschland GmbH, Germany
Javid Taheri, Karlstad University, Sweden
Christof Teuscher, Portland State University, USA
Luca Tesei, University of Camerino, Italy
Sotirios Terzis, University of Strathclyde, UK
Christof Teuscher, Portland State University, USA
Kyriakos Vamvoudakis, University of California, USA
Arlette van Wissen, Philips Research, Netherlands

Mirko Viroli, Università di Bologna, Italy
Shengxiang Yang, De Montfort University, UK
Logan Yliniemi, University of Nevada, Reno, USA

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

On The Synthesis of Adaptive Parameter-Dependent Output Feedback Controllers Through LMI-Based Optimization

Graziano Chesi

Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: chesi@eee.hku.hk

***Abstract*—This paper addresses the problem of designing adaptive output feedback controllers for stabilizing plants affected by parameters. A novel approach is proposed that allows one to design a fixed-order fixed-degree adaptive parameter-dependent output feedback controller by solving convex optimization problems with Linear Matrix Inequalities (LMIs). The proposed approach is based on the construction of a function that provides a stability margin of the closed-loop system depending on the controller. The conservatism of the proposed approach can be reduced by increasing the size of the LMIs.**

Keywords–Parameter-dependent; Adaptive Controller; Stability; LMI.

I. INTRODUCTION

Ensuring stability is of fundamental importance in engineering. Given an unstable plant, this is generally achieved by designing a stabilizing output feedback controller, i.e., a controller that elaborates the output of the plant in order to provide an input to the plant such that the so obtained closed-loop system is stable. The design of such a controller is based on the model of the plant, and several techniques can be used.

Real plants are often affected by parameters. These can happen due to various reasons. One reason is that such parameters can represent quantities that the user can modify, such as the gain of an amplifier, in order to achieve a different performance. Another reason is that such parameters can represent quantities that are unknown or subject to changes, such as the mass, resistance, temperature, etc.

Whenever the plant is affected by parameters, the output feedback controller should be able to ensure stability for all admissible values of the parameters. For this, the controller should be dependent on the parameters in general, i.e., should be able to adapt to different plants corresponding to different values of the parameters. Such a controller would be, hence, adaptive, in particular parameter-dependent.

Unfortunately, the design of stabilizing output feedback controllers for plants affected by parameters is a difficult problem. Indeed, several conditions do exist in the literature for establishing stability of systems affected by parameters, in particular conditions based on convex optimization constrained by LMIs; see for instance [1]–[5]. However, such conditions lead to nonconvex optimization problems whenever a controller is searched for, generally due to the product of the Lyapunov function and the controller that generates Bilinear Matrix Inequalities (BMIs); see for instance [6] [7]. Also, several non-LMI strategies are available for the design of stabilizing feedback controllers for plants that are not affected by parameters, however, for plants affected by parameters, such strategies could not be used due to the lack of analytical expressions (such as factorizations dependent on the parameters) or could lead to controllers of unacceptable order and degree; see for instance [8].

This paper addresses the problem of designing adaptive output feedback controllers for stabilizing plants affected by parameters. A novel approach is proposed that allows one to design a fixed-order fixed-degree adaptive parameter-dependent output feedback controller by solving convex optimization problems with LMIs. The proposed approach is based on the construction of a function that provides a stability margin of the closed-loop system depending on the controller. The conservatism of the proposed approach can be reduced by increasing the size of the LMIs. A numerical example illustrates the proposed approach. This paper extends our previous work [9].

The paper is organized as follows. Section II introduces the preliminaries. Section III describes the proposed approach. Section IV presents an illustrative examples. Lastly, Section V concludes the paper with some final remarks. This work is supported in part by the Research Grants Council of Hong Kong under Grant HKU711213E.

II. PRELIMINARIES

Notation: $\mathbb{N}, \mathbb{R}, \mathbb{C}$: sets of nonnegative integers, real numbers, and complex numbers; $\mathrm{re}(A)$: real part of matrix A; A': transpose of matrix A; $A \geq 0$: symmetric positive semidefinite matrix A; $\mathrm{spec}(A)$: set of eigenvalues of A.

Let us consider the plant

$$\left\{ \begin{array}{rcl} \dot{x}(t) & = & A(p)x(t) + B(p)u(t) \\ y(t) & = & C(p)x(t) \end{array} \right. \tag{1}$$

where $t \in \mathbb{R}$ is the time, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input, $y(t) \in \mathbb{R}^q$ is the output, $p \in \mathbb{R}^q$ is the vector of parameters, and the matrices $A(p)$, $B(p)$ and $C(p)$ are given matrix polynomials. It is supposed that

$$p \in \mathcal{P} \tag{2}$$

where $\mathcal{P}$ is the set of admissible parameters. The plant (1) is controlled by the parameter-dependent output feedback controller

$$\left\{ \begin{array}{rcl} \dot{\tilde{x}}(t) & = & \tilde{A}(p)\tilde{x}(t) + \tilde{B}(p)y(t) \\ u(t) & = & \tilde{C}(p)\tilde{x}(t) + \tilde{D}(p)y(t) \end{array} \right. \tag{3}$$

 ISBN: 978-1-61208-463-3

where $\tilde{x}(t) \in \mathbb{R}^{\tilde{n}}$ is the state of chosen order $\tilde{n} \in \mathbb{N}$, and the matrices $\tilde{A}(p)$, $\tilde{B}(p)$, $\tilde{C}(p)$ and $\tilde{D}(p)$ are matrix polynomials to determine of chosen degree $\tilde{d} \in \mathbb{N}$.

Problem 1. The problem addressed in this paper consists of determining a fixed-order fixed-degree output feedback controller (3) such that the closed-loop system (1)–(3) is asymptotically stable for all $p \in \mathcal{P}$. □

Let us observe that the plant (1) can represent the model of a nonlinear system that has been linearized for an equilibrium point of interest. In this case, the matrices in (1) are obtained by evaluating the derivatives of the vector field and of the output function of the nonlinear system at the equilibrium point and corresponding input.

III. Proposed Approach

The first step of the proposed approach is to express the closed-loop system (1)–(3) as

$$\dot{z}(t) = E(p, v)z(t) \tag{4}$$

where $z(t) \in \mathbb{R}^{n+\tilde{n}}$ is the state, $v \in \mathbb{R}^w$ is the vector of design variables in the controller, and $E(p, v)$ is a matrix polynomial in p and v. This can be simply done from (1)–(3) defining, for instance, $z(t) = (x(t)', \tilde{x}(t)')'$.

The second step of the proposed approach is to introduce a function, denoted by $\xi(v)$, that provides a stability margin of the closed-loop system depending on the controller. To this end, $\xi(v)$ could be defined under the constraint that $\xi(v) > 0$ if and only if the closed-loop system (1)–(3) is asymptotically stable for all $p \in \mathcal{P}$. Moreover, larger (respectively, smaller) values of $\xi(v)$ should correspond to more (respectively, less) stable systems. For instance, a possibility is given by

$$\xi(v) = - \sup_{p \in \mathcal{P}, \lambda \in \mathrm{spec}(E(p,v))} \mathrm{re}(\lambda). \tag{5}$$

Another possibility consists of exploiting the Hurwitz's determinants; see for instance [10]. Let us observe that one can introduce acceptable stability margins by requiring that $\xi(v)$ is greater than a specific positive value, whose definition depends on the problem requirements and on the choice of $\xi(v)$.

The third step of the proposed approach is to search for a polynomial $\zeta(v)$ that approximates $\xi(v)$ from below to a desired accuracy. This could be done by imposing

$$\begin{cases} \xi(v) & > & \zeta(v) \\ \xi(v) & < & \zeta(v) + \varepsilon \end{cases} \tag{6}$$

where $\varepsilon > 0$ is the desired accuracy.

The fourth step of the proposed approach is to search for a value of v that makes $\zeta(v)$ positive, i.e.,

$$v^* : \ \zeta(v^*) > 0. \tag{7}$$

In fact, from (6), it would follows that

$$\xi(v^*) > 0, \tag{8}$$

i.e., v^* solves Problem 1.

The search for $\zeta(v)$ satisfying (6) and the search for v^* satisfying (7) can be addressed through convex optimization problems with LMIs. Moreover, under some assumptions on the data, the conservatism of these procedures can be decreased by increasing the size of the LMIs involved.

IV. Illustrative Example

For simplicity, let us consider the plant (1) with

$$\begin{cases} A(p) = \begin{pmatrix} -1 & 0 & 1-p \\ 0 & -1 & 1 \\ 1+p & 0 & 0 \end{pmatrix}, \ B(p) = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \\ C(p) = \begin{pmatrix} 1 & p & 0 \end{pmatrix}, \ \mathcal{P} = [-2, 2]. \end{cases}$$

This plant is unstable depending on p. Indeed,

$$p = 0 \quad \Rightarrow \quad \mathrm{spec}(A(p)) = \{-1.618, -1, 0.618\}.$$

Also, it can be verified that there does not exist any stabilizing controller of order 0 and degree 0 for this plant.

Hence, we consider the problem to find an adaptive parameter-dependent output feedback controller (3) of order 0 and degree 1 that stabilizes the plant.

Let us express the matrix $\tilde{D}(p)$ as $\tilde{D}(p) = v_1 + v_2 p$, where $v_1, v_2 \in \mathbb{R}$ are the design variables. We search for a polynomial $\zeta(v)$ as described in Section III, finding

$$\zeta(v) = 0.139v_1^3 - 0.921v_1^2v_2 - 1.275v_1^2 - 0.605v_1v_2^2 \\ -0.667v_1v_2 - 1.482v_1 + 0.39v_2^3 - 2.833v_2^2 - 6.262v_2 - 4.52.$$

Hence, we search for v^* satisfying (7), finding

$$v^* = (-2, -1.660)'.$$

Therefore, we conclude that the controller (3) with $\tilde{D}(p) = -2 - 1.660p$ stabilizes the plant for all $p \in \mathcal{P}$.

V. Conclusion

A novel approach has been proposed for designing a fixed-order fixed-degree adaptive parameter-dependent output feedback controller for stabilizing plants affected by parameters. Future work will analyze its computational burden.

References

[1] P.-A. Bliman, "A convex approach to robust stability for linear systems with uncertain scalar parameters," *SIAM Journal on Control and Optimization*, vol. 42, no. 6, 2004, pp. 2016–2042.

[2] G. Chesi, "Establishing stability and instability of matrix hypercubes," *Systems and Control Letters*, vol. 54, no. 4, 2005, pp. 381–388.

[3] C. W. Scherer, "LMI relaxations in robust control," *European Journal of Control*, vol. 12, no. 1, 2006, pp. 3–29.

[4] R. C. L. F. Oliveira and P. L. D. Peres, "Parameter-dependent LMIs in robust analysis: Characterization of homogeneous polynomially parameter-dependent solutions via LMI relaxations," *IEEE Transactions on Automatic Control*, vol. 52, no. 7, 2007, pp. 1334–1340.

[5] G. Chesi, "Time-invariant uncertain systems: a necessary and sufficient condition for stability and instability via HPD-QLFs," *Automatica*, vol. 46, no. 2, 2010, pp. 471–474.

[6] H. L. S. Almeida, A. Bhaya, D. M. Falcao, and E. Kaszkurewicz, "A team algorithm for robust stability analysis and control design of uncertain time-varying linear systems using piecewise quadratic Lyapunov functions," *International Journal of Robust and Nonlinear Control*, vol. 11, 2001, pp. 357–371.

[7] F. Wang and V. Balakrishnan, "Improved stability analysis and gain-scheduled controller synthesis for parameter-dependent systems," *IEEE Transactions on Automatic Control*, vol. 47, no. 5, 2002, pp. 720–734.

[8] E. Mosca, *Optimal, Predictive, and Adaptive Control*. Prentice Hall, 1994.

[9] G. Chesi, "Robust static output feedback controllers via robust stabilizability functions," *IEEE Transactions on Automatic Control*, vol. 59, no. 6, 2014, pp. 1618–1623.

[10] G. Chesi, A. Garulli, A. Tesi, and A. Vicino, *Homogeneous Polynomial Forms for Robustness Analysis of Uncertain Systems*. Springer, 2009.

 ISBN: 978-1-61208-463-3

Multi-Layered Design and Game-Based Learning as a Pedagogical Concept

How to develop proper behavior in ARPA simulator training

Johan Hartler
Department of Shipping and Marine Technology
Chalmers Technical University
Gothenburg, Sweden
e-mail: johan.hartler@chalmers.se

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

***Abstract* — To become a professional master mariner one has to develop many different skills and have an understanding of how to act in different situations on the bridge. Within the master mariner program at Chalmers University of Technology, Sweden, simulation technologies are used to evolve pertinent skills within the educational program. A challenge with using a full scale simulator from the outset of the program is to get the students to develop both professional competencies and internalize tacit knowledge in the navigation of a ship when the interface of the simulator itself is quite demanding. By using an adaptive Multi-Layered Design approach in combination with game based learning, this paper proposes how to guide the student through a more summative learning process. The main idea is to grant limited access to what the students can do with some functions, and gradually turn on more functionality in order to develop certain experienced behaviors to get them to understand the logical approach behind selections and to make them think through why and when they should do things.**

Keywords - Simulator training; game-based learning; Multi-Layered Design; radar.

I. INTRODUCTION

There is a trend of high technical fidelity in maritime simulators. Simulators provide a rich, realistic interface with a large amount of functions and possibilities where the user can explore and experience many different scenarios. However, high-functionality interfaces are often very challenging for a user to learn [1][2]. The high amount of functions require not only a skilled interface user, but also high skills within the scope of the application if the training is to be useful and not only seen as an advanced game. The user has to be relatively experienced in the naval context in order to draw educated conclusions and solve tasks in a realistic way [3]. When a simulator is used in an educational setting, the aim is usually to teach less oriented students something about a real setting or to practice a complex activity before they perform under real conditions. Students, beginners and novices are supposed to use the simulator in this way to become more experienced [4][5]. However, the user has to possess real life experience in order to be able to fully understand the simulator and make realistic choices, but at the same time the simulator should provide exactly that – a virtual surrogate for real live experience. It would be preferable if the tacit knowledge the experienced person possess within the specific domain, could somehow be transferred to the novice.

Our underlying assumption in this study is that the knowledge of an experienced professional can be captured and taught to new students through simulator-supported learning. The goal is to convey the process of decision making and the rules for why a specific decision is made. Furthermore, the novice user should understand the underlying logic in why certain behaviors are preferable in a specific situation. The outcome is to strengthen the ability to make educated and constructive decisions and value, order and select specific important data in a large dataset. By encouraging certain behaviours in the simulator, we believe that it could be possible to transfer the knowledge of a professional maritme officer to a novice student by practicing in a guided environment. If the interface is adapted to support guided learning and certain forced behaviors, the novice could practice how an experienced mariner officer would act and understand why problems should be solved in a specific way. To achieve this, the underlying learning processes as well as the profession at hand must be understood.

The focus of this study is the interface design and game based learning as a support for teaching of professional behaviors in a maritime simulator. We suggest an adaptive simulator design where a stepwise learning approach is used, a so called Multi-Layered Design [1][6]. The functions are divided into so-called layers and adapted to the learning scenario. The choices are limited at first and then they gets less and less restricted over time. The first layer will be rather restricted from a domain point of view and train certain behaviors that should be included in the novice's basic understanding. The next layer will give slightly more freedom and the last layer should have full functionality. The complexity of the data and the technical fidelity in the simulator is not simplified. The Multi-Layered approach is then combined with gaming-based learning strategies to encourage and create enthusiasm. By using a gaming philosophy, specific behaviors can be positively encouraged through rewards. This to encourage and/or force the learner to use certain behaviors in the simulator.

The remainder of this paper is organized as follows: sections II and III will introduce the current status of today's maritime education while section IV introduce the theoretical

 ISBN: 978-1-61208-463-3

concepts further. A layered concept design is suggested and the resulting solution is discussed from a pedagogical viewpoint in section V. All design ideas are based on observations of students' behavior in the current maritime simulator at the master mariner program at Chalmers Technical University. Section VI discusses the results and presents our conclusions.

II. BACKGROUND

To become a master mariner both theoretical studies and extensive practical training are required. The practical training is initiated with simple examples on paper where different scenarios are walked through. The next step is to practice in different high fidelity simulators. When approved in the simulator the students are finally accepted to test their knowledge and ability on a real ship. To exemplify this learning process we use teaching of the Automatic Radar Plotting Aid (ARPA) in the following two sub-sections.

A. STCW MANILA 2010

STCW or the minimum Standard for Training and Certification for Watch-keeping officers describe radar navigation on management levels and is the guide for what students should know about navigation. With the start of STCW 1978, the convention has been amended several times; and the latest is STCW 2010 MANILA (used from 1 April 2012) [7]-[11]. The convention is explained in more detail in the STCW CODE, which describe competence and minimum standard of knowledge, understanding and proficiency for certification:

Competence, knowledge, understanding and proficiency

- The student should be able to show ability to use methods for demonstrating competence

Criteria for evaluating competence

- The student should use radar and ARPA to maintain safety of navigation.
- The student should show knowledge of the fundamentals of radar and ARPA.
- The student should show ability to handle the radar- and ARPA simulator plus in-service experience.
- The student should be able to interpret and analyze information obtained from radar and ARPA, taking into account the limitations of the equipment and prevailing circumstances and conditions.

The International Maritime Organization (IMO) gives out model courses with a detailed teaching syllabus to cover overall learning objectives within the convention. In this paper we focus on model course 1.07 and 1.08, [10][11] with the specific learning objectives:

Course 1.07

- 7.1 Set up and maintain an ARPA display
- 7.2 Obtain target information

Course 1.08

- 2.2 Carry out radar plotting

B. Classical approach to learning

The classical approach to learning Radar and ARPA, during the master mariner education is to start with simple scenarios and increase theire complexity during the course. The students are introduced to a two ship scenario using a "relative" motion setup to learn how to determine risk of collision. Figure 1 show this simple scenario with only two ships on a radar screen.

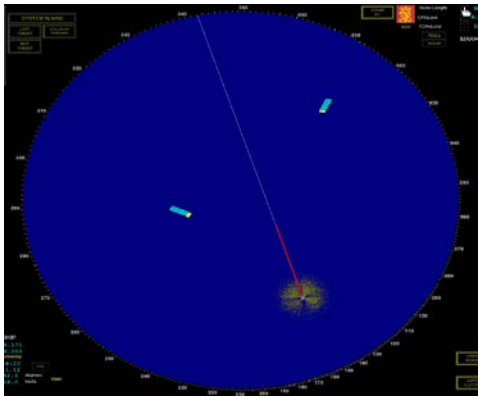


Figure 1. Scenario with only two ships on the radar

In the second scenario, there are three to four ships visible on the screen and introduces more ARPA functions that show navigational data for all the ships. In the third scenario, the complexity increases to more than seven ships and potential situations of collisions are introduced. This demands a good overall understanding of the traffic situation. Between the second and third scenario, students often switch setup from relative motion to "true" motion which can be easier to understand in more complex situations.

The left part of Figure 2 shows a part of the radar plot where six ships has been selected for tracking of position and course. More information for each ship is visualized when a ship is selected. In the right part of Figure 2 the data for two of the selected ships is visualized. Information such as speed, course, time to closest point of approach (TCPA) are shown.

Figure 2. Radar plot showing current position and surrounding ships. Six ships are plotted (numbered and with lines) and details on 015 and 003 (the two circled ships) are illustrated to the right.

With the information visualized in Figure 2 the martime officer can keep track of surrounding ships.

III. STUDY

In order to understand the learning process and how the students use the current maritime simulator to achieve proficiency, a number of activitities have been undertaken. Several groups of master mariner students have been monitored on how they use the simulator, how they develop skills over time and what type of mistakes they make. The observations have been made over four years of teaching and assessment of the course based an ARPA model course 1.07 (Operational level) and model course 1.08 (Management level) [10][11]. The participants are second year master mariner students at Chalmers University of Technology. The analysis in this paper is also made from assessment protocols from this course between the years 2010 and 2013. In the protocols, skills are given a score according to the student's performance on a 0-2 scale. Also interviews with the instructor of the course, who have more than 10 years of experience from teaching ARPA, was conducted. This was to confirm the assessment scores and get more information on identified challenges. The assessment protocols and the interview, as well as the authors own observations during these four years of teaching, lead to the same conclusions.

One of the more frequent behaviours among students are the approach of selecting all targets and using long vectors. It can be observed that in the high technical fidelity scenarios with seven or more ships the students tend to continue to select or mark all ships as the amount of ships increase in the scenarios. Figure 3 illustrates a photo taken during an assessment showing classical mistakes despite around 22 hours of practice in the simulator.

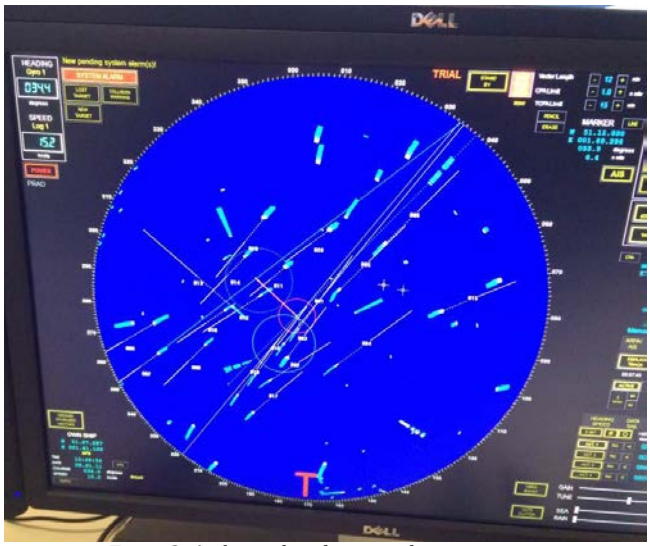


Figure 3. A photo taken during student assessment.

The students learn in the simple scenarios that all ships *can* be marked, but they do not understand the implications upon situation awareness in a more complex view, as a more cluttered radar picture is more difficult to understand (see Figure 3). It is still possible to follow all ships but that will require full attention on the radar screen, which is not a positive outcome in a real world case. On a real ship bridge, the officer of watch needs to keep control of a number of monitors and displays. Hence, the students tend to get information overload resulting in a suboptimal, bottlenecked behavior in the simulator. Best practice is to plot a maximum of 8 to 10 targets that might be of interest from an anti-collision perspective. From a teaching perspective, this creates challenges that relates to required competence in the STCW CODE for how to use plotting techniques and relative and true motion concepts, as well as setting up and maintaining multiple displays.

The different set-ups for combinations of relative and true motion vecrors poses another problem. In one interview the instructor states that "*Many students have difficulties in understanding the difference between relative and true motion, relative and true vectors, and relative and true trails*". There is up to eight different set-up combinations and all of them are appropriate for specific conditions or purposes. Relative motion shows if there is a risk of collision in an easy and quick way. True motion show real movement of vessels and are used to avoid grounding. True vectors give a hint of the direction of a vessel, and the direction is deciding which is the "stand on" vessel according to international regulations for preventing collisions at sea. Students normally do one set-up and maintain the same in all situations.

The instructor continues to discuss that "*ARPA settings and particular plotting of targets are almost impossible to teach*". This is due to the characteristics of the embedded tacit knowledge as there is never a perfect or expected setting for given conditions. The correct behavior is to switch between different settings and the officer of the watch needs to "*know*" when and why the specific setting is the right choice in that moment. Such knowledge takes time and practice to build up and it is seen amongst experienced deck officers. With better understanding and knowledge, the student would be able to change set-up according to the changing scenario. They should also be able to switch between which ship to select and track.

IV. THEORETICAL BACKGROUND TO DESIGN CONSEPT

Miller [4] was one of the first to start discussing the importance of fidelity in simulators and he made a distinction between physical fidelity and psychological fidelity. Physical fidelity is a technical aspect while psychological fidelity is how well the functional skills in the simulation relates to practice in the real world. The past years of increased computing power have largely increased the possibilities of the physical fidelity in simulations [5]. However, the ability to transfer skills from the simulator to the real world is a key element in the quality of the learning. Hence, functional fidelity is more important than technical fidelity, but a mix between good physical and psychological fidelity is needed.

The technical fidelity or the graphical user interface of a simulator is created to support a large number of complex scenarios and different types of users with different types of skills. This often powers a so-called all-in-one interface, where all various functions are visible at once [12][13][14], and so is also the case with the current maritime simulator. In

a learning situation, this could lead to confusion but also mistakes when solving specific task.

To try to guide the learning of functions we suggest a Multi-Layered Design [1][6] combined with positive behavioral encouragement from game-based learning [15] [16]. A good behavior is then praised through rewards in the simulator leading to a practice of a "correct" behavior in different situations. The idea is to transmit knowledge of how to behave in certain situations through gaming, encouragement and limitations. Creating an, for the student adaptive interface, but for the instructor an adaptable interface (possible to alter). The following sub-sections will present these concepts further.

A. *Game-based learning*

It is possible to use gaming strategies in educational settings in several ways, but it is important to differ between Game-based learning and gamification [15][16]. Gamification means the use of game characteristics to achieve something else. An example is the usage of collecting points in a commercial advertisement campaign. Game-based learning means the use of games for educational purposes.

Rules define how a player interacts in a game and is more important than the educational theme [16]. One example is chess, if one, by mistake, touches a piece and it is moved out of position, both players restore the state of the game. If a player touches a piece and then regrets the start of the move, the player must move that piece. The difference between the two situations is explicitly agreed on and understood by chess players. The formal rules say that a touched piece must be moved and still we see this extra non-written rule.

Linderoth [15] argues two ways to use rules in games for learning. The first way is used for drill training. Instead of learning mathematics from a traditional book, an example is given as a space ship game (Matteraketen/The Math Rocket) [17]. The players need to shoot down meteorites to survive but the ammunition is only refilled by solving mathematical equations. The rule then states that a player needs to learn how to solve mathematical problems to refill ammunition. Observations have shown that players might cheat and bring a friend who is good at mathematics to solve the questions.

The second way to use rules is where the rules are representing the learning itself. An example is given in [16] a simple game for understanding environmental sustainability (Harvest about sustainability). In the game, fifty fish are in the ocean and five teams will fish for ten days. Every day the teams write down on a piece of a paper how many fish they will land that day. The instructor chooses randomly to hand out fishes to each team. If there are not enough fish that day in the ocean compared to what a team wished for, no fish are handed out. Every day the amount of fish in the ocean is doubled with up to maximum 50 fish. Normally all teams choose too much on the first and the second day for the ocean to be repopulated in a sustainable way. Rules are directed towards a discussion on the population level of fish for sustainable fishing. The players learn directly from the rules of the game.

A problem with a game-based learning approach is if the player end-up in bad state of a so-called gamer mode. A gamer mode is a state when a learner uses different options from given rules in a game and carry over this behaviour into the real world. This can be problematic when the learner tries to win instead of adhering to intended learning outcome. Frank [18] describes this behavior in a model illustrated in Figure 4. The left part of the figure is the scope of the game and to the right is reality. The overlapping part between these two areas is the intended learning zone that is aimed for when using the game as a learning platform in a simulator. When in gamer mode, the player tend to focus on things in the game that are only beneficial to the game and not to the real world scenario.

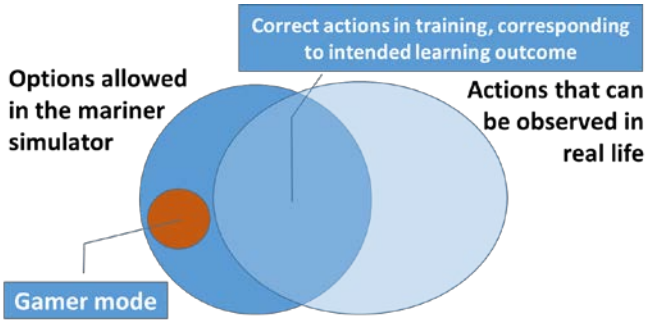


Figure 4. A model of learning outcome when a player focus on the wrong part of the game [18].

Frank [18] documents two contributing factors leading to a gamer mode. The first factor is when the game does not fully match the real world functions, corresponding to only partial functional fidelity. This invites the learner to leave the professional mode and uses the rules from the game to win. The second factor is the game-design itself. Playing for victory points does something to the learner. In a study, Frank [18] showed how the health indicator are lower at the end of the game when playing for points compared to when not playing for points. The learner enters into a mode where fulfilling game goals becomes more important than the intended learning outcome of managing military troops. Educational simulation consists of three elements according to Aldrich [3]; games, pedagogic, and simulation. Simulation enable experimentation, practice and transfer of practice to real world knowledge. The game element might not by itself contribute to intended learning outcomes but can be used to enhance learning experience. The game element contributes to engagement, enjoyment and can be a way to assess or direct the learners focus on a particular *thing*. The pedagogical element includes the meta-game with background, scenario and intended learning outcome for specific knowledge. Scaffolding is central in the learning process according to Aldrich [3] and Säljö [19].

B. *Multi-Layerd Design*

Multi-Layered Design as a concept was first introduced by Shneiderman et al in 1998 [1] and further investigated in a number of studies [6][20]-[24]. In Multi-Layered Design, the graphical user interface is divided into layers and has a sequence in which the layers are ordered [6]. The sequence

should provide a meaning to the interface structure and could be based on domain knowledge, frequency of use, or the user's task capability. It should be noted that although an application might have layers and a component like structure, it does not necessarily have a meaningful order and will not provide a proper layered design.

A layer is a set of specified functions constituting a part of an application. The layer could hold one or several functions, like program instructions, wizards, forms, data, graphical decorations and representations, or text information. Each layer always has a specific purpose, which for example could be to train a specific type of tasks (for example selecting ships on a radar plot) [1][6]. How the layer is composed depends on the intended purpose, the number of available functions, intended sequence and the level of complexity of the application at hand.

It is possible to choose if the layered design should affect only how the functions are divided or if the graphical presentation of the interface objects should change between the layers as well. For example, if a function should be visible in all layers but not available in lower layers. How to present information could also be varied between the layers.

The concept of layered design is not new, similar ideas have been used within games, learning environments, and access systems for a long time but then with different names, like levels, tiers, parts or paths. There are many types of structures and many varieties of applications using a design similar to the layered design.

C. *Adaptive and adaptable interface*

When creating layered structures, the design of the interface can follow two types of interactive approaches, either adaptable or adaptive. In an adaptable interface the users have control over the layer contents while the adapting interface is intelligent and change the contents based on external rules or algorithms [2]. Each of those include different techniques for how to actually design the graphical interface. The adaptive concept, for example, includes techniques like intelligent interfaces, self adapting menus and task based adaptation. The adaptable concept encompasses customization and user aware choices. An application can be both adaptable and adaptive for different types of users, depending on their access rights in the system.

V. RESULTS - WANTED LEARNING APPROACH

Based on the observations and the experiences from assessments and interviews, a concept for simulator learning is created. A wanted learning approach for the student is to understand the foundation of why decisions are made and what might be the consequences of a bad decision. The simulator should encourage a correct behavior and train the student in behaving like an experienced master mariner. However, this requires tacit knowledge to be transferred from the experienced mariner to the student via the simulator. A correct learning behavior, in the more advanced scenario with a large amount of ships, would be to understand how to sort the traffic information on the radar screen and select a "correct" number of ships to track. The decision should be made based on the risk of collision. Ships in close proximity to the student's own ship might require a course alteration or other activity to reduce the risk of collision. The amount of ships in more complex scenarios and reality are typically, somewhere between 15-25 ships (or more) in moderately busy European and Asian waters, depending on the scale used in the radar system (see real world example in Figure 5).

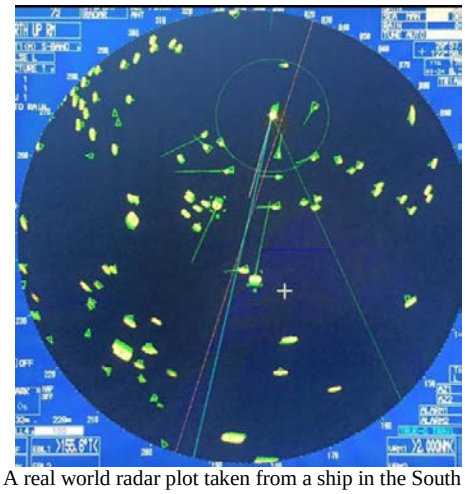


Figure 5. A real world radar plot taken from a ship in the South China sea.

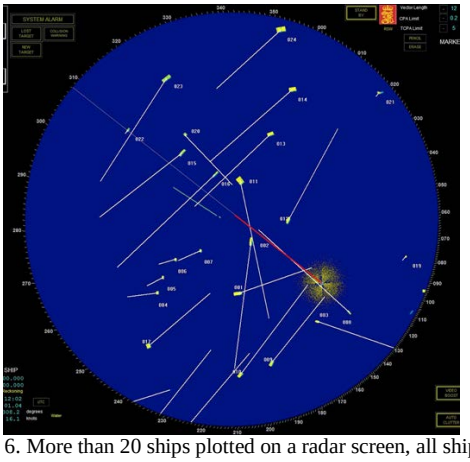


Figure 6. More than 20 ships plotted on a radar screen, all ships are selected for detailed information, creating an information bloat.

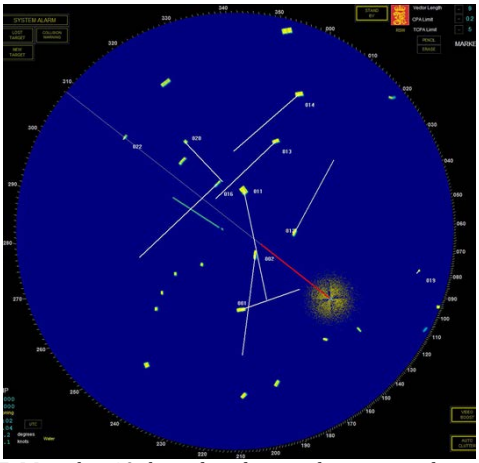


Figure 7. More than 20 ships plotted on a radar screen, only six targets selected for detailed information.

Normally, students should actively switch between a radius of 12 miles (up to 25 ships) and 6 miles (10-15 ships) on the screen. It is not possible to follow all 25 ships with full data coverage turned on, as seen in Figure 3 earlier and in Figure 6 below. Such a behavior will result in information overload. This indicates that an educated selection of ships has to take place. Figure 7 visualizes the same radar plot as in Figure 6 but with only six ships selected. It is now possible to get a quick overview of the situation and only information needed is visualized in detail and highlighted. Understanding this type of educated selection is what the students should strive for in the scenarios. To try to bridge the transfer of knowledge for how to make educated selections a concept based on Multi-Layered Design combined with game-based learning is suggested.

A. *Learning approach with a layered and guided design*

A simulator with a layered design implemented creates possibilities for a new set of training scenarios. The teaching can focus on radar plotting and how to safely navigate the ship. The training scenario, keeping in mind that training is scaffolding the knowledge, has a specific purpose and an intended learning outcome. The technical rules built into the scenario can be used to limit functions in the simulation to only allow usage of best practice combinations.

From the instructor's point of view, the layered design is adaptable and possible to customize for each intended learning scenario, while from the student's point of view the interface is adapting to how they behave in the simulator. The focused sequence is based on behavior in different situations and each layer targets to train a specific behavior. The fidelity will not change - only the rules for how to use the functions. As a first suggestion a design with three layers is chosen. The number of three layers is based on intended behavior tested during assessment.

Layer 1 - The first layer should hold only the most basic functions needed to be able to navigate but with full fidelity. The complexity of the radar plot should be realistic and the functions guide or force the students behavior. The interface should not allow the student to make unprofessional selections. If the student tries to make a selection or use a function that represents an unprofessional choice the simulator should give hints to why this is undisirable. From a graphical point of view this means that the functions should be grouped and ordered in a meaningful pedagogical manner. Functions not available in this layer should not be visible at all, since that could cause confusion. The rules for this layer should add a limit of ten targets to select. This may force the learner to prioritize early and to build experience about what type of targets that are of interest. The second learning effect should be to cancel the selection of targets that are not of interest anymore in order to be able to select and view new targets.

Layer 2 - The second layer should have less constraints for how the functions can be used. Game-based learning is used to encourage the student to make correct choices. If the student shows correct behavior and good strategies when solving problems, the simulator should be rewarding. A reward could be, for example, hints for the next upcoming risk that will help the student to make the next choice by an early warning. Another such reward could be to acknowledge correct selections and praise the student for good behavior as a feedback on earlier choices.

Like in the game "matteraketen" (math rocket) [17] the student has to safely navigate a moving vessel over an ocean, the learner needs to solve how other vessels move. This is done with the use of relative- and true-motion techniques. The second layer should help the student to practice how to switch between these modes and understand when and why to switch.

Layer 3 - The third and final layer should be very similar to what the simulator looks like today. Full fidelity, full functionality and the students have to make decisions based on previous training. The student should by now know from earlier layers how to behave. The game-based learning could still be used to give positive feedback or to improve on details and skills that are more advanced.

An example of how to map training scenarios to the layers can be seen in TABLE I. Three different steps for training are suggested, matching the layered layout.

TABLE I. PROPOSED NEW TRAINING

Scenario	# ships	Ship on collision	Layers	Game feature
1-2	10-15	1	Relative motion. 6 ships to plot True-motion available only for 1 minute	Points for correct ships plotted
3-4	20	2	8 ships to plot. Relative motion gives a hint	Hidden information
5-6	20	4	Full functionality	

The first two scenarios introduce the simulator and how to perform basic plotting. The focus is on which ship to select for more information and to follow its course. The idea is to identify the ships with potential risk of collision within the surrounding noise. Relative motion is available and true motion is available for short time spans to train the student in switching in-between those two modes. Note that the number of visible ships on the screen is rather large. The functional fidelity of the simulator has not been simplified, the scenario should mimic a real world case.

In the third and fourth scenario (used in the second layer), the number of visible objects on the radar plot is higher and more activity is required to avoid collision. Also more functions are available and can more decisions can be made. When showing correct behavior, the student is rewarded with hints that will help solve the task at hand.

The two last scenarios are played out with full fidelity, complexity and functionality. The student should now be able to handle the full simulator and based on previous training be able to demonstrate correct behaviors. This layer can also be used for exams where the student shows their skills during assessment. The interface will guide the learner through correct behaviors and give information during the time to why this was good behavior. In debriefing after the

 ISBN: 978-1-61208-463-3

simulator session it is possible for the instructor and the student to reflect and discuss their experiences and why this is or should be the best practice. The suggested design should take the student away from gamer mode and with help of limitations encourage correct behavior. Our suggestion is to use the technical rules as layers to hinder behavior not corresponding to actions amongst professionals.

VI. DISCUSSION AND CONCLUSION

It can be concluded that maritime simulator training can adopt knowledge from other domains and it is argued that some challenges in ARPA training can be partly solved with elements from Multi-Layered Design and game based learning theory. The use of technical rules can steer towards intended learning outcomes.

The first challenge is how to provide knowledge for the setup of the ARPA display in a professional manner. As shown there is different mental loads when interpreting displays with all targets plotted or only a few targets. From a professional mariner's perspective two or three different settings might be workable in those scenarios while the rest are inadequate.

The rules of the layered structure are very important when creating the layers in the suggested design. In a classic Multi-Layered Design the functions are unavailable and the division of layers are based on the number of functions per layer divided by knowledge level or how often a function is used. In the current case all functions are available. Instead the layers are based on how a function is allowed to be used. The layers are divided based on the rules of the learning scenario. Rules in the simulation should steer the learning process according to the intended learning outcome so an improved overall learning amongst learners can be accomplished. By forcing the learner to adapt to the rules of how to use different functions in each layer a proper professional behavior is practiced. A disadvantage of a game based design could be that the learner tries to optimize the behavior in the simulator to solve the problem using added game features like hidden information just to win without actually understanding the learning scenario or the intended learning outcome. An active instructor is a way to get qualitative learning and avoid this risk for gamer mode behavior in the simulation according to Frank [18].

REFERENCES

[1] Shneiderman, B. Designing the User Interface-Strategies for Effective Human-Computer Interaction. 3rd ed. 1998, Longman, Essex, United Kingdom: Addison Wesley.

[2] Dix, A. Finlay, J. Abowe, G.D. and Beale, R. Human-Computer Interaction. 3:d ed. 2004, Harlow, United Kingdom: Pearson & Prentice Hall.

[3] Aldrich, C. Learning by Doing, Section The Three Essential Elements to Successful Educational Experiences: Simulations, Games and Pedagogy.Page 79-96. Publisher Pfeiffer year 2005 San Francisco

[4] Miller. R.B. Psychological considerations in the design of training equipment. ; 1953. Report No.: WADC-TR-54-563.

[5] Glavin R&MNJ. Low- to high-fidelity simulation – a continuum of medical education? MEDICAL EDUCATION. 2003; 37: p. 22-28.

[6] Christiernin, L.G. "Layered Design Concepts, Case Studies and Processes – Theories and Implementations" Doctoral thesis, Department of Computer Science and Engineering, Chalmers University of Technology, Gothenburg University, Göteborg, Sweden, 2007.

[7] International Maritime Organization. IMO. [Online].; 2015. Available from: http://www.imo.org/MediaCentre/PressBriefings/Pages/02-WMD-launch.aspx#.VUsfDc4wyfQ [cited 2015 05 02].

[8] International Maritime Organization. STCW 1978. [Online].. Available from: http://www.imo.org/OurWork/HumanElement/TrainingCertification/Pages/STCW-Convention.aspx [cited 2015 04 25].

[9] International Maritime Organization. STCW 1978. [Online].. Available from: http://www.imo.org/OurWork/HumanElement/TrainingCertification/Pages/STCW-Convention.aspx [cited 2015 04 25].

[10] International Maritime Organisation. Radar navigation, radar plotting and use of ARPA. 2010th ed. London: IMO; 1999.

[11] International Maritime Organization. Radar, ARPA, bridge teamwork, search and rescue. 2010th ed. London: IMO; 1999.

[12] McGrenere, J. Baecker, R.M. and Booth. K.S. An Evaluation of a Multiple Interface Design Solution for Bloated Software. in The Conference on Human Factors in Computing Systems (CHI2002). 2002. Minneapolis, Minnesota, USA.

[13] Mozeico, H. A Human/Computer Interface to Accommodate User Learning Stages. Communications of the ACM, Computing Practice, 1982. 25(2): p. 100-104.

[14] Kaufman, L. and Weed. B. Too Much of a Good Thing?: Identifying and Resolving Bloat in the User Interface. in The Conference on Human Factors in Computing Systems (CHI'98). 1998. Los Angeles, USA.

[15] Linderoth J. Why gamers don´t learn more, a ecological approach to games as learning environment. Journal of gaming & Virtual Worlds. 2012;: p. 45-62.

[16] Linderoth J. Det regelstyrda lärandet. In R.Säljö ALA&. Lärare i den uppkopplade skolan. Malmö: Gleerups; 2014. p. 173-195.

[17] Matteraketen. 1994. Levande böcker (1994). Matteraketen [PC-spel]. Levande böcker: Stockholm.

[18] Frank A. Gamer mode. PhD Thesis. Stockholm: KTH Royal Institute of Technology, School of computer science and communication, Dept. of Media Technology and interaction design; 2014. Report No.: ISSN 1653-5723.

[19] Säljö, R. (2000). Lärande i Praktiken (Learning in Practice), Studentlitteratur, Lund, Sverige.

[20] Shneiderman, B. Promoting Universal Usability with Multi-Layer Interface Design. in Conference on Universal Usability. 2003. Vancover, British Columbia, Canada.

[21] Christiernin, L.G., "Guiding the designer: A Radar Diagram Process for Applications with Multiple Layers," Interacting with Computers, vol. 22(2), pp: 107-122, March, 2010.

[22] Christiernin, L.G. and Martin, A "A multi-layered aesthetical web-portal interface for governmental integration issues". In Proc. of the International Conference on Advanced Visual Interfaces (AVI 2010), pp: 341-344, Rome, Italy, Maj, 2010.

[23] Christiernin, L.G. Bäckman, R. Gidmark, M. and Persson, A. "iLayer: MLD in an Operating System Interface" In Proceedings of International Working Conference on Advanced Visual Interfaces (AVI2006), pp: 87-90, Venezia, Italy, May, 2006.

[24] Andersson, M. Heldal, R. and Christiernin, L.G. "Applying Multi-Layered Design Aspects in a Role-Based Application – An Industrial Case Study", In Proceedings of the International Conference in Human Computer Interaction, pp: 87-90, Phoenix, USA, November, 2005.

 ISBN: 978-1-61208-463-3

A Model for Experience-Based Agent Specific Trust

Mats Neovius
Faculty of Natural Sciences and Technology
Åbo Akademi University
Turku, Finland
Email: mneovius@abo.fi

Abstract — **Contemporary computerized tasks increasingly depend on inherently inaccurate information provided by autonomous agents. Reasons for the information inaccuracy are many, including the uncertainty in measurement (calibration errors), the process inherent inaccuracy (rounding), changing level of quality and, when humans are involved, differences in preferences (bias), (un)intentional violation of expectation to mention a few. Parameterization of this inaccuracy is demanded for prompt and justified adaption. Frequently, this parameterization is overlooked when models for reasoning on the inaccuracies are presented. In this paper, we address the parameterization of inaccuracy by an experience-based model. The model is based on Dempster-Shafer theory of evidence that relies on a history of experiences of subjective satisfaction on some provided data. From this history, the model derives the warranted belief and certainty that may justifiably be placed on the acquired data. The model facilitates decay and abstraction of a subset of history to a versatile score. This paper's contribution is in showing the experience-based model's generality and versatility by mapping it to EigenTrust, Subjective Logic and probabilistic trust management models.**

Keywords- Experience-based trust; multi-agent; evicence theory; adaptive systems.

I. INTRODUCTION

Collective intelligence, collaborative intelligence, participatory sensing and many related concepts share the idea of a set of decentralised autonomous agents interacting for a cause. This cause, whatever it may be, is realised as a resource that the provider(s) possess(es) and the consumer desires. Realistic examples of such resources include a measurement of a sensor, information an agent is willing to share. In such a setting, the level of trust the consuming agent may justifiably place on an acquired resource is inherently incomplete. This is due to the continuously changing context in which the resource was established, i.e., deviation in calibration, changes in temperature, mood, bias, time etc. Thus, though the provider would willingly and rightfully (untampered) share a resource, it may still be perceived by an agent as an unreliable provider in context. These inherent inaccuracies are acknowledged in participatory sensing [1] and as the parameters of quality of context by Buchholz et al. [2] as: precision (accuracy), probability of correctness (unintentional errors, e.g., bugs), *trustworthiness*, resolution (granularity, rounding) and up-to-dateness (age of measurement). They define trustworthiness as "how likely it is that the provided information is correct" [2] and as a parameter that the consumer evaluates on the provider. For the consumer (hereafter trustor) to evaluate the level of trust in a context on the provider (hereafter trustee), the history of experiences may be utilised.

An experience, as considered in this paper, is a first-hand posterior (subjective) evaluation by the trustor on a resource provided by the trustee in a proposition. The set of first-hand experiences is an agent's experience history. When combining several agents' experiences, the level of trust becomes reputation-based; a concept originally coined by Barber [3]. In reputation-based trust, referral experiences are used as witnesses to augment the first-hand experiences. Thus, reputation-based trust calls for trust transitivity and a means to discount the referrals' experiences. Further, combining the history of experiences on a system's global level provides a reputation of "what is generally said or believed" [4] about the trustee. This global view assumes a ground truth to exist that all trustees agree on. Thus, we model an experience as a posterior subjective evaluation by the *trustor* on the *trustee* at a *datum* in a *proposition* by a *score*. This four tuple view excluding the datum is shared by the FIdes REputation (FIRE) model [5]. Characterising for agent specific trust relying on experiences is that initially, in the absence of experiences, the level of trust should be vacuous. A vacuous level is the level of full uncertainty. The level of uncertainty is sometimes called the level of confidence [6]. We consider all experiences to increase certainty (decrease uncertainty); research not agreeing on this view exists [7].

In this paper we parametrize inaccuracy in a computationally light experience-based general trust model. The model features learning to trust and means to build and maintain a level of trust within group of agents (agent societies) [8]. It was originally developed for deriving the level of momentary trust on continuously changing, subjective and inherently inaccurate data with varying quality [9]. The view is sketched in Figure 1, which is inspired by the sentient object model by Fitzpatrick et al. [10]. In Figure 1, an agent may consume resources of other agents or inquire agents as referrals. The acquired resources may be weighted by the momentary discounted level of trust the trustor has in the trustee and the level the trustee claims in the resource. If triggering an actuator, the posterior trust level forms the basis for an experience. If providing the level of trust in a trustee further to another agent, this agent acted as a referral. On this view, this paper outlines the model for

 ISBN: 978-1-61208-463-3

storing, inquiring, delegating, decaying and abstracting the referrals provided experiences preserving a sense of privacy. The contribution of this paper is that we show the mapping of the general model to well-known trust management models from the autonomous agent systems point of view.

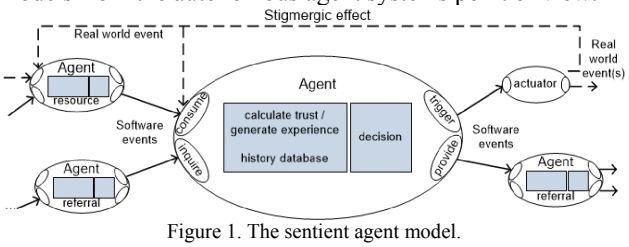


Figure 1. The sentient agent model.

The outline of this paper is as follows: in Section II we provide the motivating background and present related work. Following this, Section III introduces the foundation for the general experience-based trust model from a multi-agent perspective. Section III also defines the general model, of which a variant has successfully been implemented on in-house temperature data [9]. Section IV provides the contribution of this paper in the form of motivating the generality and versatility by mapping it to other computational models. Finally, Section V imposes a critical discussion on the findings and concludes the paper.

II. BACKGROUND AND RELATED WORK

Traditionally, trust in the context of computers related to authorisation of agents by security policies to access resources. This is an example of *policy-based trust*, a concept originally introduced by Blaze et al. [11] as a variant for specifying such security policies of a resource in terms of credentials and relationships for authorisation, also known as resource access trust [12] and credential based trust [13]. Fundamental for this is that these policies are effectively Boolean predicates and can be modelled mathematically within frameworks [13] [14] [15] providing an *absolute* level of trust in a proposition. For example, with respect to file-access rights, an agent may be trusted on a partially ordered discrete scale of $none \leq read \leq read/write$ to the extent of *read* stating absolutely that this agent may not write the file under any circumstances. Implementations of policy-based trust include access control and firewall rules. However, as this paper considers agent specific trust for setting the level of reliance on a resource by experience, rather than access to it by policies, this paper will not consider policy-based trust as such. Interested readers are directed elsewhere [16].

Autonomous agents may in contemporary open systems be either software agents or human agents. Examples of such systems include Multi-Agent Systems and the "things" in the Internet of Things initiative. Regardless of the type of the agent, the benefits are in agent collaboration. This collaboration implies a sense of mutual trust between the interacting agents. However, in the set of agents providing a resource, the consuming agents need to “know which interaction partners to trust and how to select them” [17]. Moreover, as the preferences of the trustor may be subjective, or the behaviour of the trustee may change, the computational trust management system needs to be adaptive in providing the momentary level of trust. For this, experience-based trust models relying on authentication of the agent and its behavioural history recorded as experiences may be used. Related work on similar means only considering the first-hand experiences and the global reputation include TRAVOS (Trust and Reputation model for Agent-based Virtual OrganisationS) [6].

Implementations of experience-based trust models may be centralised or distributed. In a centralised environment dedicated agents gather all experiences making the level of experience-based trust representing the collective ‘belief’, ‘doubt’, ‘evidence’ or ‘support’ that the trustee will perform in accordance with the collective’s shared expectations. Examples include online auction sites such as Ebay, product review sites such as Epinions, and discussion forums (Slashdot’s karma), to mention a few. In centralised systems, the score type is typically uniform, e.g., in Ebay {-1, 0, 1}, and the proposition the "item to be as described". For such, research on forgiveness and regret in social online settings evaluating, among others, the EVENT with respect to the agent’s reputation have been researched elsewhere [18].

In a distributed system, where each agent stores its own possibly subjective experiences, there is no global objective level of trustworthiness. For the agents to acquire second-hand trust levels (using referrals), the trust score level needs to be uniform. They are frequently partially ordered and often totally ordered [19]. Existing representations include {0, 1}, {-1, 0, 1} with $-1 \leq 0 \leq 1$, any real in [0, 1], {*low*, *med*, *high*} with $low \leq med \leq high$. Related work often overlooks the merger of a set of such scored experiences or provides a level with semantics such as "the greater the better" or a probability [20]. Such models work well when assuming that each agent has an objective level of trust and the model's task is to figure this out [21], e.g., determining if a dice is biased by repeated testing that is a stochastic processes for which statistical and probabilistic model checkers have been developed.

In an open system assuming biased agents with non-uniform preferences, varied aspiration levels and possible performance changes in the producers, stochastic processes do not suffice. In such settings, a momentary and agent specific level of trust is reasonably sought. On this, related literature has applied logical reasoning on (i) binary and discrete values, (ii) fuzzy on continuous values, (iii) transitivity and (iv) probabilistic reasoning on a value range. Existing implementations of these include (i) summation [22] [23], (ii) Regret system [24], (iii) PageRank [25], and (iv) Bpdf [26] [27] [28], EigenTrust [29] respectively. Thus, open systems demand an agent specific versatile model considering the subjectivity, means to store and share levels of trust while preserving the referral’s privacy. Moreover, as of the limited evidence, the trustor’s level of (un)certainty need to be parameterized; with the initial level of ‘no certainty’. This level of trust is dynamic, emergent, incomplete, relative, subjective and decentralised making the experience based trust systems very hard (if not impossible) to define formally [13].

Computational models for such a level “aims at supporting a decision making by computational agents in the

presence of unknown, uncontrollable and possibly harmful entities and in contexts where the lack of reliable information makes classical techniques useless" [11]. Common to such models are the demand of good quality inputs. To the best of our knowledge, general models capturing the fundamentals of trust calculations have been considered by Krukow et al. [13] [30]. TRAVOS [6] model considered a very similar view, however, omitting decay or discounting of second-hand trust levels that they correctly note to be truth tellning.

III. EXPERIENCE-BASED TRUST

Experience-based trust is frequently defined in line with Gambetta [31] as a subjective probability between two individuals, also called 'reliability trust' [4] describing the probability an agent *A* expects agent *B* to deliver. However, we consider experience-based trust as a parameter supporting a decision with a sense of relative security. For example, having an infinite resource of single coloured balls *ball* ∈ {*red*, *green*, *blue*}, the posterior reliability trust indicate the reliance that the next picked ball is of a specific colour. From this, assuming even distribution, a utility function defining the rationality of the decision can be defined.

To capture this, we use the broader definition of trust, called 'decision trust' [4] similar to that of McKnight and Chervaney [32] with the extension that the trustee is any identifiable matter [33], def. 1: Trust: "*The extent to which a trustor is willing to depend on a trustee in a given situation with a feeling of relative security, even though negative consequences are possible*". This definition implies that trust is relevant only when something can go wrong, that trust is proposition specific and that it is a metric describing the relation of warranted reliance a trustor places on the trustee. Let this relation be denoted by $_{\mathrm{T}}$. Moreover, consider the definition's situation as the proposition ζ that defines the exclusive and exhaustive outcomes of a frame of discernment, i.e., a trust relation with a level ω in a proposition ζ between *A* and *B* is denoted $A_{\zeta \mathrm{T} \omega} B$. The definition also underlines the need of *uncertainty* as opposed to certainty, where uncertainty for "*do not know*" must not be confused with distrust of "*do not trust*" [14]. Thus, the definition calls for a metric where increased uncertainty in an experience indicate a decrease in the weight of the evidence. This view enables a *decay* of experiences without subverting the foundational meaning of the experience, e.g., by time as a function of forgiveness or regret [34].

A. General properties of a trust relation

The most important property of a trust relation is the *unique identification* of the counterpart. The arity have been defined as a one-to-one, one-to-many, many-to-many or many-to-one [12] where many is an identifiable set of trustees. Other properties outline that a trust relation is 1) subjective, 2) asymmetric, 3) incomplete, 4) evolves over time, 5) proposition specific and 6) transitive (with restrictions). Below we briefly discuss 6), directing interested readers elsewhere [35].

The trust relation's transitivity is frequently disputed, i.e., if $A_{\zeta \mathrm{T} \omega} B$ and $B_{\zeta \mathrm{T} \omega} C$ does this imply that $A_{\zeta \mathrm{T} \omega} C$? Related literature examines this problem in greater detail [12] [36] [37]. For this paper we consider trust relations transitive over a chain of 'positive' decision trust propositions, i.e., if ω indicates a level for an affirmative decision, then $A_{\zeta \mathrm{T} \omega 1} B \wedge B_{\zeta \mathrm{T} \omega 2} C \Rightarrow A_{\zeta \mathrm{T} \omega 3} C$. For ω indicating distrust, this is considered not to hold as it would require deciding whether or not your enemy's enemy is your friend [38]. Hence, trust is in this paper considered transitive, but distrust is not [39].

B. The Experience(s)

For representing an experience and the history of such, we follow Krukow's [13] and Teacy et al. [6] general models. We consider an experience a four tuple and utilise Dempster-Shafer theory view of subjective probabilities. An element of this tuple is the score that should be accurate enough to have semantics, and at the same time general enough to map to computational methods. As the score type is subject to the used computational method and this paper is about a general model, we present a score type only as proof of concept when the model is mapped to other methods. Moreover, to meet with the property of incompleteness and that trust evolves; a means for decay per experience is introduced. We stress that this decay must not subvert the experience, merely degrade its weigh.

An experience *Exp* is the manifestation of an agent's (trustor) posterior subjective evaluation score $\eta \in \{<score>\}$ of an observation on a trustee $\delta \in \{< agents >\}$ at datum ϵ in a proposition $\zeta \subseteq \{< proposition >\}$. We represent this as a four tuple $(\delta, \epsilon, \zeta, \eta)$, e.g., an experience by trustor $P \in \{<agents>\}$ at datum $\epsilon \leq \epsilon_0$ where ϵ_0 is the momentary datum in proposition ζ with score η is denoted $Exp^P(\epsilon) = (\delta, \epsilon, \zeta, \eta)$. The history of an agent *P*'s experiences $Exp^P(\epsilon_i)$ for i = 1, …, n is a set of such four tuples, i.e., $\{(\delta, \epsilon_i, \zeta, \eta)\}$. Thus, adding a new experience $(\delta, \epsilon_0, \zeta, \eta)$ to the history $\{(\delta, \epsilon_i, \zeta, \eta)\}$ is straight forward $Exp^P(\epsilon_j) = (\delta, \epsilon_0, \zeta, \eta) \cup \{(\delta, \epsilon_i, \zeta, \eta)\}$ where j = 1, ..., n, ϵ_0. The datum may be virtually any continuous matter or composition of such, but often considered time. Projections on this four tuple is possible. That is, $Exp^A(B, \epsilon)$ defines the projection on agent *A*'s experiences on (B, ϵ) and similarly for other projections, i.e., $Exp^A(B, \epsilon) = \{(B, \epsilon, \zeta, \eta)\}$ and $Exp^A(B, \epsilon_p, x) = \{(\epsilon_i, \eta)\}$ for $x \subseteq \zeta$ and $\epsilon_i \leq \epsilon_p$ for i = 0, 1, … p. Thus, for a specific datum ϵ_i the projection returns a singleton assuming that an agent cannot interact in several matters simultaneously. In addition, we note the deliberate loose definition of δ, that with this notation may be a group of agents, supporting the trust relation's arity.

C. Type of Score

The general model's score type must be versatile. For this, we propose the score type of a tuple (*sat*, *unsat*) as for satisfactory and unsatisfactory where *sat*, *unsat* ∈ [0, 1] and *sat* + *unsat* ≤ 1. Subadditivity is fundamental for expressing uncertainty and decay without subverting the semantics of the experience, i.e., the level of certainty is *sat* + *unsat* where theoretical full certainty is *sat* + *unsat* = 1. Moreover, coarsening a multinomial proposition $|\zeta| \geq 3$ to a binomial $|\zeta| = 2$, this binomial score type is applicable on any $|\zeta| \geq 2$,

e.g., let *ball* ∈ {*r*, *g*, *b*}, deriving {*r*, *b*} is {*r*} + {*b*} + {*r*, *b*} where '+' denote sum of *sat* and *unsat* respectively.

With score η, an experience is $(\delta', \epsilon_i, \zeta, (sat, unsat))$. Related work considering a similar score type include Noorian et al. [40] and methods using Beta probability density functions. The type's semantics incorporate that of Dempster-Shafer theory, i.e., experience certainty is captured by the combined weight and distribution by relative weigh. A vacuous experience may thus be expressed as (0, 0), a dogmatic experience (the probabilistic view) is when *sat* + *unsat* = 1, and absolute experiences (binary or Boolean view) when (*sat*, *unsat*) = (0, 1) or (*sat*, *unsat*) = (1, 0). Thus, a score *sat* = 0.3 and *unsat* = 0.5 is valid indicating a certainty of 0.8. From this, uncertainty *u* is easily derived, $u = 1 - sat - unsat$ as is the dogmatic expectation value of satisfyability as *sat* / (*sat* + *unsat*).

D. Decay of Experience

Decay of an experience relates to forgetting or forgiveness. When the datum is time, it is natural to weigh recent experiences over older. Let the decay factor be λ where $0 \leq \lambda \leq 1$. This factor relies on a continuous datum ϵ by which it decays. We write d_{ϵ_m} for the general decay function *d* at datum ϵ_m on an experience $Exp^{\delta}(\epsilon_i)$ where $\epsilon_i \leq \epsilon_m$ as:

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

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

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

Here each experience is decayed by λ, defining the speed of 'forgiveness'. The closer λ is to 1, the slower the speed with λ = 1 indicating no decay motivated when consistency is assumed. Dually, λ = 0 indicate complete decay, motivated when the experiences are random. Hence, the effect of decay is that an experience score η is reduced by factor λ with respect to the datum, i.e., η at ϵ_m is less or equal to η at ϵ_n when $n \leq m$ and $\lambda \leq 1$. Realistically, ϵ may be time.

E. Abstracting Experiences

To calculate with a set of experiences, a composition to an abstract experience, denoted *Abs* is necessary. This abstraction is done by some datum, say ϵ_m, hence Abs_{ϵ_m}. The composition of decayed experiences outlines a momentary decayed score, the abstracted score *absscore*. We define this for $\epsilon_n = 1, \ldots, m$ and $\epsilon_n \leq \epsilon_m$:

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

Thus, $Abs_{\epsilon_m}\left(Exp^{\delta}(\delta', \epsilon_m, \zeta)\right) = (absscore)$. We define *absscore* as a tuple (*abssat*, *absunsat*) where *abssat*, *absunsat* ∈ $\mathbb{R}^+$. The semantics of this is linear: "the greater the more certain". Updating the *absscore* is recursive [41] whenever λ is universal, continuous and applied on all experiences locally.

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

Here $Exp^{\delta}(\delta', \epsilon_{m'}, \zeta, \eta)$ is the new experience. In case no new experience was recorded at $\epsilon_{m'}$, $Exp^{\delta}(\delta', \epsilon_{m'}, \zeta, \eta)$ = (<vacuous>), i.e., (0, 0). Thus, abstraction is irreversible and provides a sense of privacy that decay enhances on.

IV. The General Model Mapped to Existing Computational Methods

In the subsequent sections, we will show how the general model may be mapped to a probabilistic view.

A. Probabilistic views

Semantically a purely probabilistic view is very rich. From the *absscore* expectation value this is directly derived by *abssat* / (*abssat* + *absunsat*). However, the probabilistic view abstracts (assumes) certainty, i.e., the expectation value outcome is equivalent for Beta (4, 2) and Beta (12, 6) where obviously, the latter should indicate higher certainty. Hence, for the probabilistic view to be reasonable, certainty is complete, i.e., it is a dogmatic view. We can see this as a valid approach only for statistical modelling. In addition, the presented general model also captures consistent behaviour, e.g., assume there to be an event of 0.7 probability of success, then by setting λ = 1, *absscore* will approach the 0.7 relation between *abssat* and *absunsat*. On such an event, the model holds as decay does not subvert the expectation value. Thus, we conclude that the probabilistic view can be expressed by the general model.

B. Discrete views

A discrete view is one where the level of trust is expressed in a countable space. This space is a set that is typically totally ordered, e.g., none ≤ small ≤ large. Thus, as probabilistic views are possible, this less expressive discrete view on a totally ordered set is possible to express by the general model as well.

C. EigenTrust

EigenTrust [29] is an algorithm originally targeted for Peer-to-Peer systems that computes a global trust value for an agent. The algorithm requires each experience to be rated either satisfactory or unsatisfactory, making the score binary $\eta \in \{0, 1\}$. Such experiences may be modelled by the general model, and merged to the *absscore*. Thus, EigenTrust function on the abstracted score $scr_{\delta\delta'}$ of agent δ on δ' as $scr_{\delta\delta'} = abssat_{\delta\delta'} - absunsat_{\delta\delta'}$. A score $scr_{\delta\delta'}$ is normalized with respect to $scr_{\delta\delta''}$ where $\delta'' \in$ {<agents>} \ δ, i.e., with respect to agents δ have recorded direct interaction with whenever $scr_{\delta\delta''} \geq 0$, otherwise $scr_{\delta\delta''} = 0$. These normalized values is the $scr_{\delta\delta''}$ vector that when organized as a global *I*-by-*J* matrix *M* denoting on one row the level of trust an agent perceives in the other agents. When *M* is transposed M^{T}, a row denotes the level of trust others' have in an agent. Hence, multiplying M^{T} by itself is as if asking friends, i.e., $(M^{\mathrm{T}})^2$. Obviously, this assumes transitivity, and as the score is positive, only positive

transitivity. For $(M^{T})^{n}$ where n is sufficiently large, the matrix rows will converge within some tolerance providing an unweighted the global objective trustworthiness vector. Thus, the general model maps to EigenTrust.

D. Subjective Logic

Subjective Logic (SL) is a probabilistic logic providing a means for transitivity and derives a level of subjective belief in an entity in a proposition developed primarily by Jøsang [27] [38]. It is related to Dempster-Shafer theory and consists of a set of well-defined logical operators on its basic type *opinion* ω being a four tuple (b, d, u, a) as for belief, disbelief, uncertainty and base rate. This opinion represents a binomial view, i.e., one where the exclusive and exhaustive frame of discernment polarity is 2, for and against. SL on dogmatic opinions (no uncertainty) falls back on traditional probabilistic logic and functions as Boolean logic when the opinions are absolute.

For the opinion to capture the general model's *absscore* overlooking the level of (un)certainty, a non-informative prior weigh parameter W is introduced. The assignment of W is delicate depending on the frequency of new experiences with respect to the datum and level of decay making it application specific. With this, the expectation value is defined *abssat* / (*abssat* + *absunsat* + W) implying that W guarantees incompleteness in form of non-additivity. A mapping function from *absscore* including W to the opinion type basing on the abstracted history of experiences have been presented by Jøsang [27] [41]:

$$\omega \begin{cases} b = \frac{abssat}{abssat+absunsat+W} \\ d = \frac{absunsat}{abssat+absunsat+W} \\ u = \frac{W}{abssat+absunsat+W} \\ a = base\ rate \end{cases} \Leftrightarrow \begin{cases} abssat = \frac{Wb}{u} \\ absunsat = \frac{Wd}{u} \\ a = base\ rate \end{cases} exp. \quad (5)$$

The SL is also related to a Beta probability density function (Bpdf) [28] as *absscore* maps to the Bpdf input tuple (α, β). Hence, visualising the SL as a Bpdf is possible. For a vacuous initial view $i = 0$ of $Abs_{\epsilon_m}\left(Exp^{\delta}(\epsilon_i)\right) = (0, 0)$ to be a horizontal Bpdf, the non-informative prior weigh W need to be 2 and the base rate 0.5. Thus, we conclude that the general model maps to opinions of SL.

V. Discussion and Conclusion

To capture the uncertainty of 'do not know' for something unanticipated, we have presented a general model for experience based trustworthiness relying on Dempster-Shafer theory of evidence. More pragmatic studies on the application of this are found elsewhere [9] [42]. As $|\{(\delta, \epsilon, \zeta, \eta)\}|$ is finite, $|absscore| < \infty$ holds whereas the evidence on any binomial view is incomplete voiding the traditional probabilistic views of Markov chains or Monte Carlo simulations. In addition, the well-known shortcoming of Dempster's rule of combination producing counter-intuitive results in case of strong conflict has been resolved [43]. This is also in line with Pearl [44] stating that "belief theory is a theory on the probability of provability as opposed to probabilities of truth". In addition, fuzzy logic operating on crisp measures about linguistically vague and fuzzy propositions is different from the presented model operating on uncertain but on crisp propositions [38] [45].

This general model of trust presented in this paper parameterises the level of reliability placed on a trustee in a proposition by disjoint experiences. The model has been implemented on real data in related work [9]. This paper shows how to abstract these experiences to a composite score and how this score may be mapped to well-known methods. Thus, the contribution of this paper is in motivating the generality and versatility of the experience-based trust model and the specific score type. Further facilitating the use of an experience-based model alike is its computational lightness featuring decay by some datum.

Acknowledgment

This research is funded by the Academy of Finland project "Merge: Merging digital hydraulic Systems and Supercomputing: New possibilities to improve productivity, reliability and service for hydraulic machines" (Grant nr. 286094).

References

[1] S. Kanhere, "Participatory Sensing: Crowdsourcing Data from Mobile Smartphones in Urban Spaces," in 12th IEEE Int. Conf. on Mobile Data Management, pp.3-6, 2011.

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

[3] B. Barber, The Logic and Limits of Trust, Rutgers University Press, 1983.

[4] A. Jøsang, R. Ismail and C. Boyd, "A survey of trust and reputation systems for online service provision," Decis. Support Syst., vol. 43, no. 2, pp. 618-644, 2007.

[5] T. Huynh, N. Jennings and N. Shadbolt, "An integrated trust and reputation model for open multi-agent systems," Autonomous Agents and Multi-Agent Systems, vol. 13, no. 2, pp. 119-154, 2006.

[6] W. Teacy, J. Patel, N. Jennings and M. Luck, "TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources," Autonomous Agents and Multi-Agent Systems, vol. 12, no. 2, pp. 183-198. , 2006.

[7] Y. Wang and M. Singh, "Evidence-based trust: A mathematical model geared for multiagent systems," ACM Trans. Auton. Adapt. Syst., vol. 5, no. 4, p. 28 pages, 2010.

[8] S. Sen, "A comprehensive approach to trust management," in Proc. of the Int. Conf. on Autonomous Agents and Multi-Agent Systems, pp. 797-800, 2013.

[9] M. Neovius, "Adaptive Experience-Based Composition of Continuously Changing Quality of Context," in Int. Conf. on Adaptive and Self-Adaptive Systems and Applications, 2015.

[10] A. Fitzpatrick, G. Biegel, S. Clarke and V. Cahill, "Towards a

 ISBN: 978-1-61208-463-3

Sentient Object Model," in Workshop on Engineering Context-Aware Object Oriented Systems and Environments, 2002.

[11] M. Blaze, J. Feigenbaum and J. Lacy, "Decentralized Trust Management," in Proc. of the IEEE Symposium on Security and Privacy, 1996.

[12] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," IEEE Communications Surveys and Tutorials, vol. 3, no. 4, pp. 2-16, 2000.

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

[14] M. Carbone, M. Nielsen and V. Sassone, "A Formal Model for Trust in Dynamic Networks," in Proc. of Int. Conf. on Software Engineering and Formal Methods, 2003.

[15] S. Weeks, "Understanding Trust Management Systems," in Proc. of the IEEE Symposium on Security and Privacy, 2001.

[16] D. Artz and Y. Gil, "A survey of trust in computer science and the Semantic Web," Web Semantics, vol. 5, no. 2, pp. 58-71, 2007.

[17] S. Keung and N. Griffiths, "Trust and reputation," in Agent-Based Service-Oriented Computing (Chap. 8), pp 189–224, Springer, 2010.

[18] A. Vasalou, A. Hopfensitz and J. Pitt, "In praise of forgiveness: Ways for repairing trust breakdowns in one-off online interactions," International Journal of Human-Computer Studies, vol. 66, no. 6, pp. 466-480, 2008.

[19] M. Neovius, "Trustworthy Context Dependency in Ubiquitous Systems," TUCS dissertations nr. 151. PhD thesis, Turku, Finland, 2012.

[20] S. Ruohomaa, L. Kutvonen and E. Koutrouli, "Reputation Management Survey," in Int. Conf. on Availability, Reliability and Security, pp. 103-111, 2007.

[21] P. Massa and P. Avesani, "Trust Metrics on Controversial Users: Balancing Between Tyranny of the Majority and Echo Chambers," Int. J. on Semantic Web and INformation Systems, vol. 3, no. 1, 2007.

[22] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," in Proc. of the 33rd Hawaii Int. Conf. on System Sciences, 2000.

[23] J. Schneider, G. Kortuem, J. Jager, S. Fickas and Z. Segall, "Disseminating Trust Information in Wearable Communities," Personal Ubiquitous Computing, vol. 4, no. 4, pp. 245-248, 2000.

[24] J. Sabater and C. Sierra, "Social ReGreT, a reputation model based on social relations," SIGecom Exch., vol. 3, no. 1, pp. 44-56, 2001.

[25] L. Page, S. Brin, R. Motwani and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Technical Report. Stanford InfoLab, 1999.

[26] S. Buchegger and J.-Y. Le Boudec, "A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks," in P2PEcon, 2004.

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

[28] L. Mui, M. Mohtashemi and A. Halberstadt, "A Computational Model of Trust and Reputation for E-businesses," in Proc. of the 35th Annual Hawaii Int. Conf. on System Sciences, 2002.

[29] S. Kamvar, M. Schlosser and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in Proc. of the 12th Int. Conf. on World Wide Web, 2003.

[30] K. Krukow, N. M. and V. Sassone, "Trust models in ubiquitous computing," Philos Transact A Math Phys Eng Sci., no. 366, pp. 3781-3793, 2008.

[31] D. Gambetta, "Can We Trust Trust?," in Trust: Making and Breaking Cooperative Relations, Chapter 13, Department of Sociology, University of Oxford, 2000, pp. 213-237.

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

[33] C. Castelfranchi and R. Falcone, "Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification," in Proc. Int. Conf. on Multi Agent Systems., 1998.

[34] S. Marsh and P. Briggs, "Examining Trust, Forgiveness and Regret as Computational Concepts," in Computing with social trust, Chapter 2, Springer, 2009, pp. 9 - 43.

[35] Z. Yan and S. Holtmanns, "Trust Modeling and Management: from Social Trust to Digital Trust," in Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions, IGI Global, 2007.

[36] B. Christianson and W. Harbison, "Why isn't trust transitive?," in Proc. of the Security Protocols Int. Workshop, 1996.

[37] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in Proc. of the 2nd Asia-Pacific conf. on Conceptual modelling, 2005.

[38] A. Jøsang, Subjective Logic, http://folk.uio.no/josang/papers/subjective_logic.pdf, visited 24.11.2015.

[39] T. DuBois, J. Golbeck and S. A., "Predicting Trust and Distrust in Social Networks," in IEEE Third Int.l Conf. on Privacy, Security, Risk and Trust, 2011.

[40] Z. Noorian, S. Marsh and M. Fleming, "Multi-layer cognitive filtering by behavioral modeling," in Int. Conf. on Autonomous Agents and Multiagent Systems, 2011.

[41] A. Jøsang and R. Ismail, "The beta reputation system," in Proc. of the 15th Bled Conference on Electronic Commerce, 2002.

[42] M. Neovius, M. Stocker, M. Rönkkö and L. Petre, "Trustworthiness Modelling on Continuous Environmental Measurement," in Proc. of the 7th Int. Cong. on Environmental Modelling and Software, 2014.

[43] A. Jøsang and S. Pope, "Dempster's Rule as Seen by Little Colored Balls," Computational Intelligence, vol. 4, no. 28, pp. pp. 453-474, 2012.

[44] J. Pearl, "Reasoning with Belief Functions: An Analysis of Compatibility," Int. J. of Approximate Reasoning, vol. 4, no. 5/6, pp. 363-389, 1990.

[45] A. Jøsang, "A logic for uncertain probabilities," Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol. 3, no. 9, pp. 279-311, 2001.

 ISBN: 978-1-61208-463-3

Adaptive Spatio-Temporal White Space Sensing in Multiple Antenna

Chang-Jun Ahn
Graduate School of Engineering
Chiba University
Chiba, Japan
e-mail:junny@m.ieice.org

***Abstract*—Cognitive radio is an emerging approach to implement efficient reuse of the licensed spectrum by detecting unoccupied spectrum bands and adapting the transmission to those bands while avoiding the interference to primary users. However, rigorous requirements are put on the white space sensing that the secondary user should have the ability to detect the primary signal in very low signal to noise ratio (SNR). To achieve the requirements, in this paper, we discuss the adaptive white space sensing for cognitive radio system based on generalized likelihood-ratio test over Rayleigh fading channel in multiple antennas. The test statistics are based on softly combined spatio-temporal diversity to enhance the detection performance. Additionally, we give the expression of the asymptotic detection performance.**

Keywords–white space sensing; spatio-temporal diversity; cognitive radio; generalized likelihood-ratio test.

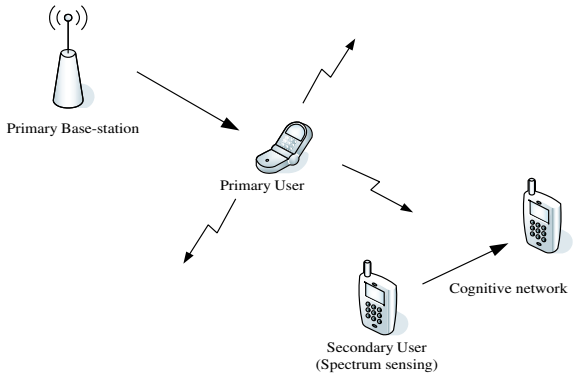


Figure 1. The concept of cognitive radio networks.

I. INTRODUCTION

Spectrum sharing remains one of the most important goals for wireless communication systems. Until this time, the principle has been to assign exclusive frequency bands to different systems or different operators, and systems that used adjacent frequency channels were required to use appropriate spectrum masks to avoid harmful interference with each other. In this case, the spectrum utilization is very limited.

Cognitive radio is an emerging approach to implement efficient reuse of the licensed spectrum by detecting unoccupied spectrum bands and adapting the transmission to those bands while avoiding the interference to primary users as shown in Fig. 1 [1]. This novel approach to spectrum access introduces unique functions at the physical layer: reliable detection of primary users and adaptive transmission over a wide bandwidth [2]-[5].

However, in order to avoid the harmful interference with the primary system, the cognitive radio needs to sense the availability of the spectrum (so called white space). Furthermore, rigorous requirements are put on the spectrum sensing that the secondary user should have the ability to detect the primary signal in very low SNR [6], [7]. Cooperative communication has been known recently as a way to overcome the limitation of wireless system. In some works, the cognitive radios are allowed to cooperate for sensing the spectrum, so that the issues are addressed [8]-[10]. Multiple antenna based cognitive radios are also proposed in [11]-[13].

In this paper, we discuss the adaptive white space sensing for cognitive radio system based on generalized likelihood-ratio test over Rayleigh fading channel in multiple antennas. As the transmitted data and channel impulse response are independent from each other, we can calculate the variability of space and time domains, separately. Although the detection with joint process of the spatio-temporal domains using likelihood ratio test shows the optimum property, it is difficult to give a clear expression of the test statistic. Therefore, the suboptimal method is given with spatio-temporal soft combination of individual test statistics in each domain. Additionally, we give the expression of the asymptotic detection performance. This paper is organized as follows. The white space sensing is described in Section II. In Section III, we describe the adaptive spatio-temporal detection. In Section IV, we show the computer simulation results. Finally, the conclusion is given in Section V.

II. WHITE SPACE SENSING

We assume that the cognitive radio system has been deployed with M received antennas. The channel response between the primary transmitter and the mth antenna of the secondary receiver h_m is modelled as independent and identically distributed (i.i.d.) complex random variables with variance σ_h, and $E(||h_m||^2) = 1$. When band-limited primary signal d_l is transmitted, the received signal at the mth antenna of secondary receiver can be given by

$$y_{m,l} = h_m d_l + n_{m,l}, \tag{1}$$

where $n_{m,l}$ is the additive white Gaussian noise with the variance σ_n^2. In order to avoid the harmful interference with the primary system, the cognitive radio needs to sense the availability of the white space adaptively. The goal of white space

 ISBN: 978-1-61208-463-3

sensing is to decide between the following two hypotheses:

$$\begin{aligned} \mathcal{H}_0 &: \ y_{m,l} = n_{m,l} \quad \text{(absence of primary signal)} \\ \mathcal{H}_1 &: \ y_{m,l} = h_m d_l + n_{m,l} \quad \text{(presence of primary signal)}. \end{aligned} \tag{2}$$

In the traditional multiple antennas reception, we can combine the signal on each antenna to acquire diversity gain. If h_m and $n_{m,l}$ are known to the secondary receiver, maximum ratio combination (MRC) can be used to maximize the SNR. However, in general, h_m, d_l, and $n_{m,l}$ are unknown in the secondary receiver. Therefore, the case with unknown h_m, d_l, and $n_{m,l}$ is more practical. From this reason, it is reasonable to model d_l as complex Gaussian with zero mean and variance σ_d^2.

III. Adaptive Spatio-temporal Detection

Adaptive detection can be performed with the time-variability of h_m without the knowledge of the noise variance. Reviewing the expression in (1), we find that the variances of d_l and h_m can be transformed to each other without any changes on the distribution of $h_m d_l$. As d_l and h_m are independent from each other, we can calculate the variability of space and time domains, separately. Although the detection with joint process of the spatio-temporal domains using likelihood ratio test shows the optimum property, it is difficult to give a clear expression of the test statistic. Therefore, the suboptimal method is given with spatio-temporal soft combination of individual test statistics in each domain.

A. Spatio variability

Based on the variability of h_m, we get the generalized likelihood-ratio test (GLRT) for the binary hypothesis testing in Eq. (2)

$$\begin{aligned} \Lambda_1 \quad = \quad & ML \ln \big(\sum_{m=1}^{M} \sum_{l=1}^{L} |y_{m,l}|^2 \big) \\ & - L \sum_{m=1}^{M} \ln \big(\sum_{l=1}^{L} |y_{m,l}|^2 \big) \\ & - ML \ln (M) \quad \begin{cases} \mathcal{H}_1 & \Lambda_1 \geq \eta \\ \mathcal{H}_0 & \Lambda_1 < \eta \end{cases} \end{aligned} \tag{3}$$

where L is the available independent samples which is smaller than $2BT_c$ where B is the bandwidth, T_c is the coherence time. From the detection theory, the asymptotic distribution of Λ_1 for both hypotheses follows the central and non-central χ^2 distribution of $2M$ degrees of freedom separately. That is

$$2 \cdot \Lambda_1 \sim \begin{cases} \mathcal{X}^2_{2M} & \text{under} \ \ \mathcal{H}_0 \\ \mathcal{X}'^2_{2M}(\lambda_1) & \text{under} \ \ \mathcal{H}_1 \end{cases} \qquad ML \gg 1 \tag{4}$$

where

$$\lambda_1 = L \cdot \frac{\sigma_d^4}{\sigma_n^4} \cdot \sum_{m=1}^{M} |h_m|^4. \tag{5}$$

where σ_n is the variance of noise $n_{m,l}$.

B. Temporal variability

Similarly, conditioned on the time variability on d_l, the GLR test can be derived with the following result

$$\begin{aligned} \Lambda_2 \quad = \quad & ML \ln \big(\sum_{m=1}^{M} \sum_{l=1}^{L} |y_{m,l}|^2 \big) \\ & - M \sum_{l=1}^{L} \ln \big(\sum_{m=1}^{M} |y_{m,l}|^2 \big) \\ & - ML \ln (M) \quad \begin{cases} \mathcal{H}_1 & \Lambda_2 \geq \eta \\ \mathcal{H}_0 & \Lambda_2 < \eta \end{cases} \end{aligned} \tag{6}$$

Likewise, the asymptotic distributions of Λ_2 is

$$2 \cdot \Lambda_2 \sim \begin{cases} \mathcal{X}^2_{2L} & \text{under} \ \ \mathcal{H}_0 \\ \mathcal{X}'^2_{2L}(\lambda_2) & \text{under} \ \ \mathcal{H}_1 \end{cases} \qquad ML \gg 1 \tag{7}$$

where

$$\lambda_2 = M \cdot \frac{\sigma_h^4}{\sigma_n^4} \cdot \sum_{l=1}^{L} |d_l|^4 \ . \tag{8}$$

where σ_h is the variance of channel h_m.

C. Adapive spatio-temporal soft combination

To fully utilize the observations in space and time variability, adaptive soft combination is performed to maximize the detection performance. The test statistics after combination is given by

$$\Lambda = \omega_1 \Lambda_1 + \omega_2 \Lambda_2 \quad \begin{cases} \mathcal{H}_1 & \Lambda \geq \mu \\ \mathcal{H}_0 & \Lambda < \mu \end{cases} \tag{9}$$

where μ is the detection threshold. The accurate weight can be obtained numerically to get the optimal solution. Moreover, according to the Gaussian approximation of the $\mathcal{X}^2$ distribution (restricted to the limits on the antenna number M in a practical system, this approximation is not quit matched for small M), we have the asymptotic optimal detection with the maximal ratio processing

$$\begin{aligned} \omega_1 &= \frac{L}{\sqrt{M^2 + L^2}}, \\ \omega_2 &= \frac{M}{\sqrt{M^2 + L^2}} \ . \end{aligned} \tag{10}$$

The two test statistics Λ_1 and Λ_2 are thought to be independent from each other due to different variabilities they are base on. Moreover, different Doppler frequency and delay spread of channel show the different coherence time and bandwidth. Therefore, we need to identify the different coherence time and bandwidth. Since L is smaller than $2BT_c$, we need to choose the suitable L, adaptively. Observing Eq. (10), we can get independent optimum weights ω_1 and ω_1 by choosing the suitable L due to different channel variabilities. To derive the more compact expression of the performance, it is common practice to approximate a weighted sum of chi-square variables by that $2 \cdot \Lambda \ \sim \ \alpha \mathcal{X}^2_\xi$. The scaling factor α and the effective

 ISBN: 978-1-61208-463-3

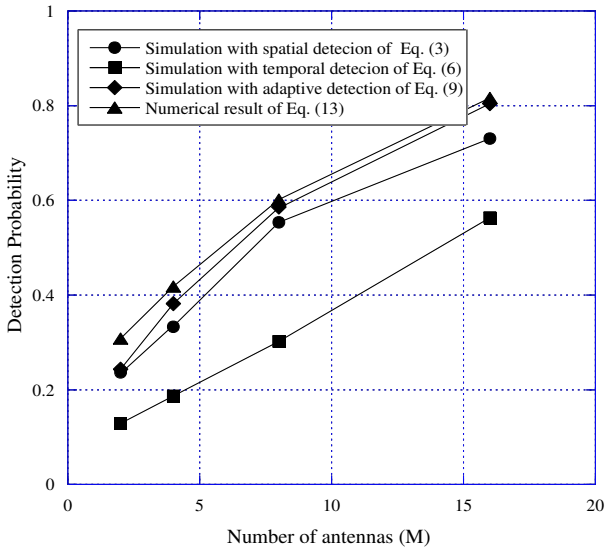


Figure 2. Detection performance for various M with $L = 50$, $P_{FA} = 0.1$ and σ_s^2/σ_n^2=-7dB.

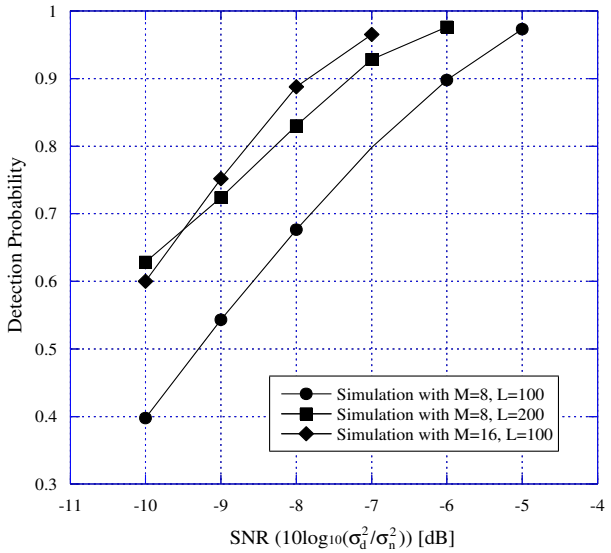


Figure 3. Detection performance for various M and L with $P_{FA} = 0.1$.

degrees of freedom ξ are chosen so that the distribution has the same first two moments as Λ. Thus we have

$$\begin{aligned} \mathcal{H}_0 \quad : \quad \alpha_1 &= \frac{M\omega_1^2 + L\omega_2^2}{M\omega_1 + L\omega_2} \\ \xi_1 &= \frac{2(M\omega_1 + L\omega_2)^2}{M\omega_1^2 + L\omega_2^2} \\ \mathcal{H}_1 \quad : \quad \alpha_2 &= \frac{(2M + 2\lambda_1)\omega_1^2 + (2L + 2\lambda_2)\omega_2^2}{(2M + \lambda_1)\omega_1 + (2L + 2\lambda_2)\omega_2} \\ \xi_2 &= \frac{((2M + \lambda_1)\omega_1 + (2L + \lambda_2)\omega_2)^2}{(2M + 2\lambda_1)\omega_1^2 + (2L + 2\lambda_2)\omega_2^2} \ . \end{aligned} \tag{11}$$

Finally, the approximate probability of false alarm P_{FA} and probability of detection P_D are given by

$$P_{FA} = Q_{x^2_{\xi_1}}(\frac{2\mu}{\alpha_1}) \tag{12}$$

$$P_D = \int_0^\infty \int_0^\infty Q_{x^2_{\xi_2}}(\frac{\alpha_1}{\alpha_2} Q^{-1}_{x^2_{\xi_1}}(P_{FA})) p(\lambda_1) p(\lambda_2) d\lambda_1 d\lambda_2 \tag{13}$$

Obviously, it is easy to get a constant false alarm rate(CFAR) detector in which the detection threshold in Eq. (9) is given by

$$\mu = \frac{1}{2}\alpha_1 Q^{-1}_{x^2_{\xi_1}}(P_{FA}). \tag{14}$$

IV. Computer Simulation Results

Fig. 1 shows the concept of cognitive radio networks to identify the white space. In this section, we consider the received antennas number M to evaluate the detection probability in the cognitive radios based on generalized likelihood ratio test. In general, due to the cost limitation, the multiple antennas communication systems have a few antennas. Therefore, the received antennas number M is less than the samples number L which is only restricted to the detection duration.

Fig. 2 shows the detection performance for various M with $L = 50$, $P_{FA} = 0.1$ and SNR=-7dB. The performance improvement by exploiting two domains of spatio-temporal diversity of the received signal is obvious when M is large ($M \geq 4$). When M is small, the approximation of GLRT is not good enough which leads to bad detection probability. Particularly, the detection performance of adaptive detection is 7% degraded compared with the numerical result when $M = 3$. Moreover, Eq. (13) is given under the assumption that the test statistics in the spatio-temporal domains are independent. Thus the actual correlation will cause larger error when the P_D is large. As a result, the detection performance of adaptive detection and numerical result is the approximately same when the P_D is large.

Fig. 3 shows the detection performance for various M and L with $P_{FA} = 0.1$. From the simulation results, the detection probability is rapidly improved by increasing the number of antennas M and the number of the available independent samples L. Particularly, the detection performance of adaptive detection is more rapidly improved with increasing M than that of with increasing L. There should be a tradeoff between M and L because of the equivalent of spatial and time domains in the detection.

V. Conclusion

We discuss the detection for cognitive radio system over Rayleigh fading channel in multiple antennas. The spatio-temporal diversity is exploited to improve the detection performance. The time variability and the spatial variability are both considered to give better detection performance. The performance improvement by exploiting two domains of spatio-temporal diversity of the received signal is obvious when M is large ($M \geq 4$). When M is small, the approximation of GLRT is not good enough which leads to bad detection probability. Particularly, the detection performance of adaptive detection is 7% degraded compared with the numerical result when $M = 3$. Moreover, the detection performance is more rapidly

 ISBN: 978-1-61208-463-3

improved with increasing M than that of with increasing L. However, the number of antennas in a practical system restricts its performance gain that it is only suitable to detect signal in a medium low SNR (greater than -10dB). As the future work, we will adapt the proposed detection method to Google project Loon.

ACKNOWLEDGEMENT

This work is supported by the Grant of Scientific Research (C) No. 26420339 from the Japan Society for the Promotion of Science (JSPS).

REFERENCES

[1] J. Mitola III, "Cognitive radio: an integrated agent architecture for software defined radio," *KTH Royal Institute of Technology*, Sweden, 2000.

[2] C. M. Cordeiro, K. Challapali, and D. Birru, "IEEE 802.22: An Introduction to the First Wireless Standard based on Cognitive Radios," *Journal of communications*, Vol.1, no.1, pp.38 47, April 2006.

[3] H. Lee, K. Han, Y. Hwang, and S. Choi, "Opportunistic band sharing for point-to-point link connection of cognitive radios," *Proc. of CROWNCOM2009*, pp.1-6, June 2009.

[4] B.S. Manoj, R. Rao, and M. Zorzi, "On the Use of Higher Layer Information for Cognitive Networking," *Proc of Global Telecommunications Conference Globecom2007*, pp.3568-3573, November 2007.

[5] W. Zhang, U. Mitra, "A Spectrum-Shaping Perspective on Cognitive Radio," *Proc of DySPAN 2008*, pp.1-12, October 2008.

[6] J. Notor, "The Evolution of Spectrum Sharing in the IEEE 802.22 WRAN Standards Process," Available at:http://www.eecs.berkeley.edu/dtse/3r_otherpapers.html

[7] IEEE 802.22 WG, "ETRI FT Philips Samsung Proposal," *IEEE docs: 22-06-0005-01-0000*, January 2006.

[8] D. Cabric, S. M. Misha, and R. W. Broderson, "Implementation issues in spectrum sensing for cognitive radios," *Proc. of Asilomar Cinference on Signals, Systems, and Computers*, Vol.1, pp.772 776, November 2004.

[9] W. Zhang, and K. B. Letaief, "Cooperative Communications for Cognitive Radio Networks," *Proceedings of the IEEE*, vol. 97, no.5, pp.878-893, May 2009

[10] X. Wang, A. Wong, and P. Ho, "Stochastic Channel Prioritization for Spectrum Sensing in Cooperative Cognitive Radio," *Proc. of Consumer Communications and Networking Conference (CCNC2009)*, pp.1-6, January 2009.

[11] E. Soltanmohammadi, M, Orooji, M. Naraghi-Pour, "Spectrum Sensing Over MIMO Channels Using Generalized Likelihood Ratio Tests," *IEEE Signal Processing Letters*, vol. 20, no. 5, pp. 439-442, May 2013.

[12] S. Kumar, S. Chouhan, "Spectrum sensing in MIMO cognitive radio with temporally and spatially correlated signal," *Proc. of IEEE International Symposium on Wireless Communications Systems (ISWCS2014)*, pp. 664-669, August 2014.

[13] S. Zin, X. Zhang, "Compressive spectrum sensing for MIMO-OFDM based Cognitive Radio networks," *Proc. of IEEE Wireless Communications and Networking Conference (WCNC2015)*, pp. 2197-2202, March 2015.

 ISBN: 978-1-61208-463-3

Autonomic Cooperation Strategies for Robot Swarms

Catherine Saunders
Supervised by: Roy Sterritt, George Wilkie
School of Computing and Mathematics
Ulster University
Northern Ireland, UK
E-mail: saunders-c2@email.ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk

Abstract— **In this paper, we describe two strategies that allow a swarm of simulated robots to cooperate. For a swarm of robots to function cooperatively, self-management and autonomy are essential. Direct communication is used to enable swarm entities to communicate. The research aim is to evaluate various architectures and protocols for cooperation strategies that enable swarm robots to ask for, and respond to requests for assistance. The work is in two phases. Only phase 1 is described in this paper. The first phase involves the creation of simulation environments of robot swarms. This phase enables us to develop, evaluate and refine suitable architectures and protocols for swarm cooperation. Using simulation, it is possible to assess swarm cooperation in the large (essentially hundreds or thousands of robots per swarm). In the second phase, the cooperation protocols developed from phase 1 will be trialed on a small number of physical robots, to evaluate the complexity introduced from the real world. The 1st architecture simulated features a hierarchy with Ruler robots communicating with a swarm; the Ruler robots can request the help of the swarm. The swarm is only able to respond to the Rulers, intra-swarm communication is not possible. In the 2nd architecture simulated, a non-hierarchical homogenous swarm is able to communicate by posting help messages to a centralized Message Board entity. In future experiments, architectures involving the incorporation of a Message Board role within each swarm robot, thus removing the disadvantages associated with having a centralized component, will be explored.**

Keywords- Robot; Autonomic Computing; Swarm; Simulation; Cooperation;

I. INTRODUCTION

The purpose of this research is to investigate cooperation strategies that will enable a swarm of robots to collaborate and perform a task without human involvement. The aim is to show how direct communication could work within a swarm scenario. Most swarm research tends to focus on indirect communication, this normally involves changing the environment to influence other swarm members, or responding to other swarm members in order to maintain distance and mimic flocking. In future, we want to compare the performance of different approaches and determine the optimal cooperation strategy for swarm collaboration. We are interested in direct communication in the form of messages sent via a central controller or by direct communication from robot to robot. We are working on creating simulations to test various cooperation strategies; this paper will focus on our current research and conclude with future research ideas.

Autonomic Computing [1] takes its name from the Autonomic Nervous system, which can maintain bodily functions independent of conscious thought [2]. The aim of Autonomic Computing is to improve the self-management of autonomous software systems. This paper is part of a research project that seeks to design a model, which will allow swarm entities to communicate information in order to collaborate as a whole. In order to do this, each entity in addition to being self-managing, must be capable of receiving and reacting to communications from other swarm members. Within Autonomic Computing, there is the concept of an Autonomic Element (AE), which consists of an Autonomic Manager (AM) and a Managed Component (MC). Having an AM that uses a feedback loop to constantly check on the state of a system is particularly applicable to a robot swarm [1]. The hardware of each swarm robot represents the MC, the AM is the software that must monitor the battery life, component state, and direct the local behavior of the robot. Robot AM's must be able to cooperate with each other in order for the autonomic swarm to function efficiently. To be truly autonomic, a system must be Self-Aware, Environment-Aware, Self-Adjusting and Self-Monitoring [3].

Space exploration is an area that could benefit from incorporating Autonomic Computing ideas. Future space missions will seek to go beyond the monolithic rover concept and instead feature multiple autonomous rovers or spacecraft. It is important that Autonomic self-management techniques are incorporated into future missions that feature multiple entities. It would not be feasible for humans to manage the actions of every member of a swarm, especially in an emergency situation. NASA's concept missions demonstrate their interest in more ambitious fully autonomous swarm exploration.

The NASA Autonomous Nano Technology Swarm (ANTS) project features a concept mission known as the Prospecting Asteroid Mission (PAM). This mission would involve sending 1000 spacecraft to explore an asteroid belt. The reason for sending a large number of spacecraft is to counter the expected large-scale decimation of the swarm [4]. The mission would include 10 scientific instruments, with each spacecraft carrying only 1 instrument; we are using this example as inspiration for our research scenario. The swarm would feature different roles, such as Ruler, Messenger and Worker. Autonomic computing ideas are essential in order to ensure that swarm entities are self-

 ISBN: 978-1-61208-463-3

managing and able to cooperate effectively with each other [5][6].

In this paper, we describe two simulations that feature a swarm of robots using direct communication in order to cooperate. Section II gives an overview of related work and the different approaches within swarm research. Section III describes the two different approaches and the C# simulations. Future work is discussed in Section IV; this will involve further simulation work in order to decrease the number of unanswered help requests, and also testing on mobile robots

II. RELATED WORK

Within swarm research there are 3 general types of systems that have been explored; centralized, decentralized and hybrid approaches. A centralized system consists of a central controller that collates data from swarm members; this enables it to intelligently co-ordinate how each swarm member should behave [7]. The disadvantages of this type of system are that it does not scale well, the larger the swarm, the less efficient the controller will be at processing information and coordinating the actions of the swarm. As there is a central controller, any damage incurred can negatively affect the behavior and performance of the swarm and may also jeopardize the overall mission [7].

In contrast to this, a decentralized system operates in a Peer-to-Peer manner with communication occurring between swarm members. A Peer-to-Peer approach helps to avoid the bottleneck that can occur when there is a central controller processing all swarm communication traffic. Another advantage of Peer-to-Peer approaches is that if swarm members are damaged, the swarm can still function, as no member is indispensable. In a centralized system, if the central controller becomes damaged, the swarm would no longer be able to function cohesively [7].

A hybrid system results from the combination of centralized and decentralized strategies to varying degrees. A hybrid system can take the form of a decentralized swarm were the communication takes place locally but also includes another supervisory element that analyzes global data and provides overall mission direction [7]. The NASA PAM swarm fits the definition of a hybrid model as it features a hierarchy of Rulers, Messengers and Workers. The Rulers would be able to coordinate the behavior of the Workers by organizing them into teams and choosing exploration targets [4].

A decentralized Peer-to-Peer model is presented in [8], where a navigation system is proposed with each robot within a swarm maintaining a table of location information of every other robot within the swarm. The robots broadcast messages with location information to their neighbors; this is then distributed throughout the swarm. To test the messaging protocol, a robot declares itself to be a target, other robots must move towards this robot using the location information they have received and stored in their table. The authors were able to test their research on physical foot-bot robots [8]. This is something we would like to do in future. Another example of broadcasting within a swarm is explored in [9], the authors created a Global Coordination System, where information is exchanged by agents within a decentralized swarm by using a wireless sensor network. Each robot starts from the same position within an indoor environment, they are able to keep track of their position by checking how far they have moved from their start position. To help the swarm search for targets, robots use the location information being broadcast by other swarm members.

There is not as much research dedicated to decentralized direct communication between members of a robot swarm. Most swarm research tends to focus on implementing a system that is either centralized or one that is decentralized and uses indirect communication techniques, such as pheromones. Indirect communication can be achieved by changing the environment in a way that influences the others that are operating in that environment. This is known as Stigmergy; it is seen in nature and is the basis of much bio-inspired research. A virtual pheromone approach has been explored in [10], robots are placed at random positions within an arena and given the task of sweeping the perimeter. A pheromone trail is left by each robot to notify others that an area has already been mapped.

For the NASA PAM mission and other future missions that involve multiple robots carrying out complex tasks, it may be necessary to equip robots with more intelligence than is present in a purely reaction based system. In [11] an interesting hybrid approach is discussed, a cluster of 3 robots begin their mission as a swarm but can change to a master/slave configuration when cooperation is required. The robots are set the task of finding an object; if a robot finds the object it assumes the role of Leader and sends the location to the other 2 robots. The slave robots cannot communicate with each other, only with the Leader, the Leader then broadcasts the data collected from both slaves.

Role switching is also featured in the SWITCH project [12], which was developed for the RoboCup competition [13]. The RoboCup is held annually and has robots competing in teams to play soccer; the aim of the competition is to improve many areas of computer science including cooperation between robots. The SWITCH robots are able to change their role from Striker to Defender in response to how the game is progressing; each role has its own goals and strategies. This idea of being able to change roles depending on the situation could prove useful, as there are benefits to both the centralized and decentralized approaches. Being able to switch between the two is a useful adaptive technique that would allow a swarm to operate under centralized control but without the disadvantages this brings.

We have not implemented roles in our current simulation research but may consider it in future as a useful failsafe mechanism. Specifically, if a swarm is routing messages via a centralized communication element, which becomes damaged, the swarm may no longer be able to cooperate. By using roles, any member of the swarm can self-nominate and become a central controller, thus the limitations of a centralized system are avoided. The self-nominated robot would cease their exploration task and change their ‘role’ to that of a centralized communication element.

 ISBN: 978-1-61208-463-3

III. CURRENT WORK

This section describes two simulations that we have designed and implemented in C#. The systems are a mixture of both decentralized and centralized approaches, in both, each swarm robot is autonomous. We use a central Message Coordinator in the 2nd simulation, which could fall prey to the dangers of a centralized system. In future we would like to incorporate the coordinator as a role into each swarm robot. A simulation is useful as it enables us to create numerous robots without being constrained by the limitations of physical hardware. To ensure rigor, it is also necessary to perform real life experiments, since unexpected results can arise from the complexity posed in the real world. The final stage of our experiments will involve testing our cooperation models using a small cluster of four Dr Robot X80-H robots [14].

In Section III-A, we describe a system that features a Ruler AE, which can communicate with every member of the swarm. Sending a message to every swarm robot AM each time help is required is not very efficient. It is simple to accomplish in a simulation but could prove more challenging with actual robots. As a result of this, we created another design with a dedicated centralized Message Board. The simulation described in Section III-B features a homogenous swarm as opposed to the hierarchical system described in section III-A. Swarm robots still cannot communicate with each other and must use a central Message Board to post Help requests. When a robot finishes its task it checks the Message Board for help requests that it could fulfill.

A. Implementation 1 – Rulers and Workers Hierarchy

Most swarm research seeks to mimic natural systems e.g., flocking, shoals, foraging; however, we are interested in direct communication and instilling more intelligence in each individual swarm robot. In this simulation, our scenario involves a robot finding an interesting feature that requires other members of the swarm to move towards its location and assist. This version was inspired by the NASA PAM project; it features 3 Ruler robots that communicate directly with a swarm. The scenario involves Ruler robots requesting help from members of a swarm when they encounter interesting terrain that the Ruler cannot traverse. Swarm robots only respond if they have not reached their target destination and begun their own experiments. When the Rulers discover an interesting feature situated within terrain that they cannot traverse, they send a message to the swarm asking for help. The Rulers can communicate with every member of the swarm but the swarm can only communicate with the Ruler robots and not with each other. The swarm only replies to help requests and cannot initiate contact with the Ruler.

The map in Figure 1 includes orange and black tiles, which represent interesting terrain features that the 3 Ruler robots encounter during the course of the simulation. The user specifies the number of robots in the purple swarm. Each robot runs in its own thread, its start position, terrain capability and mission priority are all randomly generated. The help request that is sent includes the Ruler's mission priority, location and terrain encountered (i.e., Terrain capability required). This is then used by each swarm robot to decide whether or not to respond. If a swarm robot's mission priority is higher than that of the Ruler, they ignore the help request. There are two conditions that must be met in order for a swarm robot to respond to a Ruler's help request, they must be able to cross that type of terrain specified and have a lower mission priority than the Ruler. Those that respond to the Ruler's help request are placed on a 'Helpers List', provided they are within a certain distance of the Ruler robot. In Figure 1, the green and blue Rulers are flashing a proximity beacon; only responding robots within this beacon area are added to the 'Final Helpers List'. The Rulers then send a confirmation message to each robot on the 'Final Helper List'. The swarm robots that receive the confirmation message only move towards the Ruler robots location if they have not reached their destination and begun experiments.

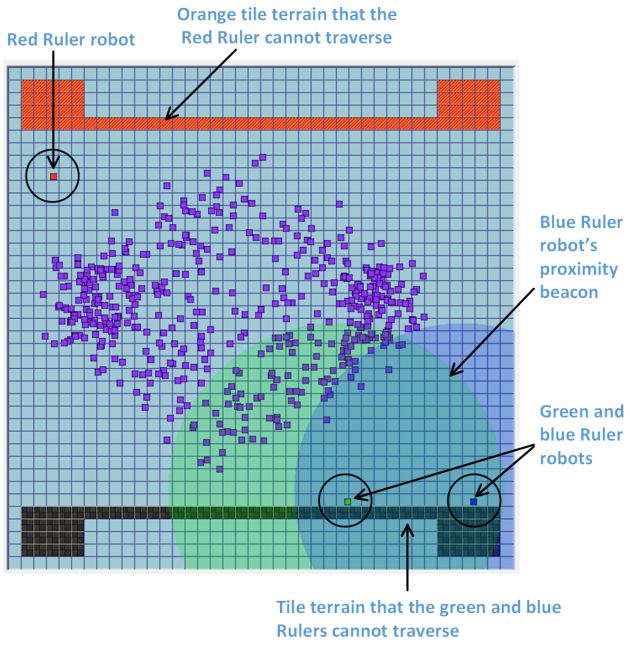


Figure 1. Ruler robots communicating with a swarm, a tile map was used with the orange and black sections representing areas that the Ruler robots could not traverse.

The flow diagrams in Figure 2 and Figure 3 show the processes (or protocols) of a Ruler robot and a swarm robot. The Ruler robot AM differs from the swarm robot AM, yet each is similar in that they both contain a continuous self-checking loop. In Figure 2, the Ruler robot creates a Helper List with the information of each swarm robot that responded to the help request. It then iterates through this list and places robots that are located within the beacon area on to the 'Final Helper List'.

 ISBN: 978-1-61208-463-3

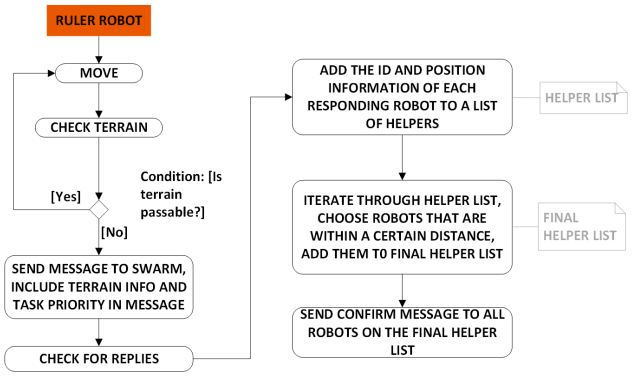


Figure 2. Ruler robot protocol

The swarm robots have an internal self-managing loop that checks for messages and formulates responses based on mission data. If the message that is received from a Ruler states a mission priority that is less important than the swarm robot's own mission priority, it will choose to ignore the message. In this respect, the swarm robots do possess some personal autonomy within the system. They may determine that their task is more important and that by refusing to help they will ultimately be benefitting the swarm mission objectives.

In Figure 3, the flow diagram (protocol) shows the loop used by each swarm robot, they only respond to messages, which have a higher mission priority than their own. If they respond to a message and do not receive confirmation from the Ruler, they continue to move to their original target destination.

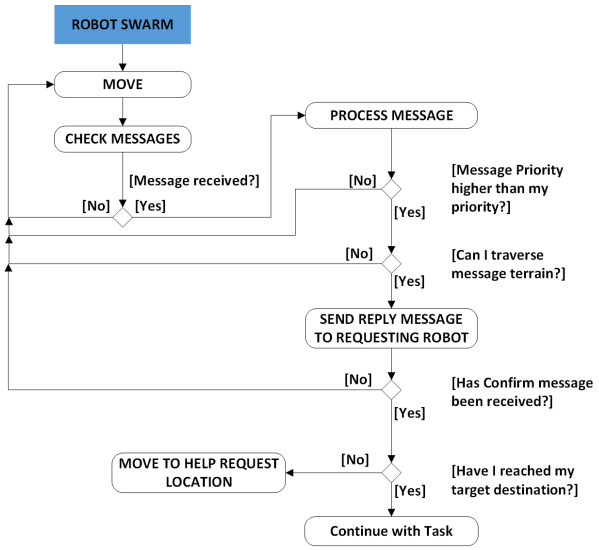


Figure 3. Swarm robot flow protocol

The system as a whole is self-optimizing, it can adapt and change its behaviour based on the terrain encountered. Rulers act as Autonomic Managers of the swarm component, they can utilize this component when necessary. The Ruler robot has the capability to reconfigure the swarm component by drawing their resources when an interesting feature is discovered. Ruler robots can reconfigure the swarm behaviour by sending a help message and choosing helpers, the system is therefore capable of self-optimizing when certain situations arise. The Ruler robots only act as Rulers in certain situations as having a swarm explore autonomously improves the chances of finding interesting features. We are using impassable terrain as a problem that necessitates help from the swarm, but this is merely a scenario to facilitate collaboration. Future work may look at using a foraging task instead where help is needed from other swarm robots to successfully forage an interesting item.

In this implementation there are 3 designated Rulers. However, in future we would like to design the swarm so that anyone can self-configure and become a Ruler when they find interesting terrain or features. Being able to change and enhance one's capability dynamically would be more autonomic and fulfill the self-configuring aspect of the Self-CHOP paradigm.

B. Implementation 2 – Message Board and Swarm

In this version, there are no Ruler robots, the swarm entities co-operate by posting their help requests to a Message Board AE. The Message Board is not a centralized controller; it is a passive tool that is used by the swarm to coordinate tasks, it does not provide global task direction. The Message Board is a different type of AE in comparison to the Robot AE. Within the simulation it does not exist as a visible entity, the AM is tasked with storing help requests.

In a real life experiment, the Message Board may exist as a satellite that can communicate with all robots. The Message Board could also be a swarm robot that is static and dedicates all of its power and resources to enabling communication within the swarm. The assumption being that it would be less taxing for a swarm robot to send a message to a designated static swarm robot than every other robot in the swarm. A failsafe mechanism would be to allow any swarm robot to take on the role of the Message Board. If the static Message Board swarm robot was damaged or destroyed, another robot could then nominate itself and change its 'role' from 'Explorer' to 'Message Board'. This would be more autonomic and allow the system to suffer significant losses yet still function.

The Map in Figure 4 mostly consists of green tiles that represent normal traversable terrain. The blue water tiles are used to represent an interesting feature that needs to be explored by the swarm. The simulation shown in Figure 6 features a swarm of red and yellow robots, each running in their own Thread and with randomly generated capabilities. Each swarm robot is given either a Red or Yellow color to represent a different scientific instrument; this is inspired by

 ISBN: 978-1-61208-463-3

the NASA PAM mission, which features up to 10 instruments. In future we would like to add more instruments/colors to enrich the scenario. The start and target coordinates are also randomized, as is the speed at which each robot moves. The speed is used to work out how much battery power the robot has used; each robot starts the simulation with 100% battery.

If a robot reaches the water tiles, it sends a help message to the Message Board with details of its location, the battery life required to complete exploration, and the type of scientific instrument required. There are two instruments, red and yellow; if the robot sending the help request is a red robot, then it would need the help of a yellow robot. In the simulation, a yellow robot will always request help from a robot with a red instrument, and vice versa. The message includes the Requesting robot's ID, Instrument required, and Battery Life needed to complete the exploration task. The Message Board entity receives and stores all help messages from the swarm.

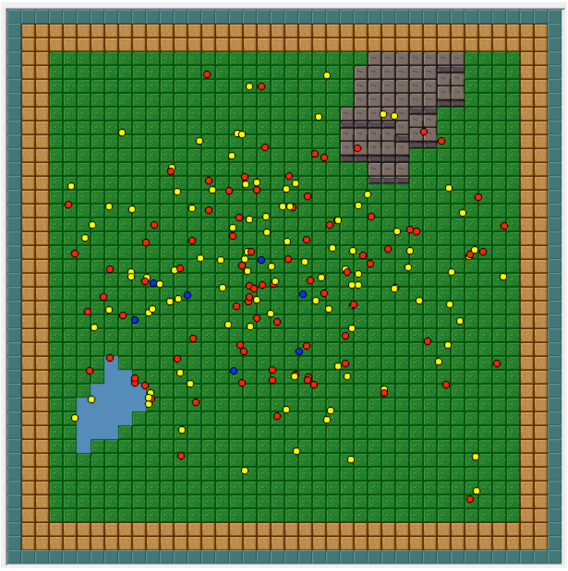


Figure 4. Swarm and Message Board simulation with terrain map. Red and yellow robots represent different instruments. Blue robots are those that are responding to help requests.

When a robot reaches its target location and finishes its task, it enters an idle state; it then asks the Message Board for a tailored list of all unfulfilled help requests. With this request, it includes its current location and instrument type. The Message Board checks the location coordinates provided by the idle robot and filters out help requests from robots outside a certain distance. It also removes any requests that do not require the idle robot's instrument type, the tailored list is then sent to the idle robot. The idle robot then chooses the help request that requires the least amount of battery power. This has led to a number of help requests not being fulfilled due to the battery life required being higher than the battery capacity of any available idle robot. Future research will look at choosing a request with the shortest time to completion window as this will help avoid a selfish swarm scenario.

When an idle robot chooses a help request, it sends a message to the Message Board and waits for confirmation. The Message Board checks that the help request is still available and unanswered, if it is, then it marks the help request as 'Completed' and sends a confirmation message to the idle robot. The idle robot then changes its color to blue and moves to the location in the help request. Unlike Version 1 of the simulation, where many robots could fulfill a help request, in this version, only one robot can respond to a help request. This is a feature that could be changed so that the requesting robot specifies in the help request how many helpers it needs for a given task.

The flow diagram in Figure 5 shows the decision-making processes performed by an idle robot. When a robot becomes idle, it asks the Message Board entity for a list of current unfulfilled help requests. The idle robot pauses until it receives a response from the Message Board, the response is nearly immediate and the wait time does not negatively impact the swarm behaviour. However if a swarm is very large, a single Message Board element might experience lag when trying to process list requests from the swarm. A bottleneck could occur and result in a large number of idle robots waiting for responses from an overtaxed Message Board. Future work will address whether it is necessary to have multiple Message Board nodes if the swarm is very large. The Message Board flow diagram in Figure 6 shows the processes followed when a Help Request is received, and also when an idle robot requests a tailored Help Request list.

The simulation requires further work in order to reduce the number of help requests that go unanswered. Currently, robots are able to choose the help request that requires the least amount of effort. Future work will incorporate a time to completion window and instruct swarm robots to respond to the request with the least time remaining.

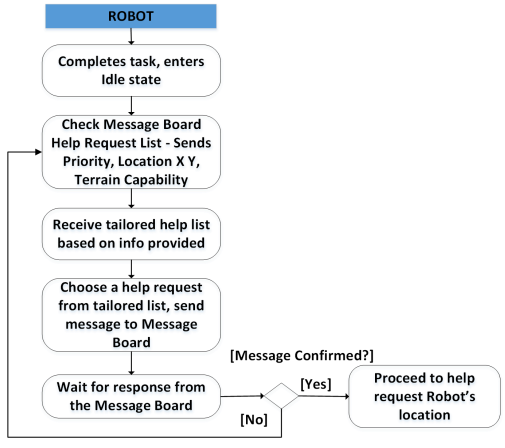


Figure 5. Robot decision-making flow diagram

 ISBN: 978-1-61208-463-3

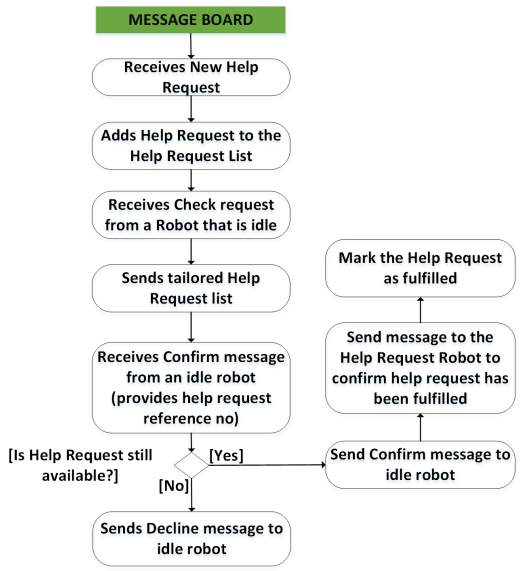


Figure 6. Message Board responding to a help request

IV. FUTURE WORK

Section IV-A describes features that could be added to the simulation in order to improve the cooperation between swarm entities. Section IV-B discusses the final stage of our research, which will involve testing the cooperation strategies on physical mobile robots.

A. Simulation Experiments

Our current approach involves a swarm of robots and a Message Board entity, however if the Message Board is damaged, the swarm has no way of communicating. Another approach would be to design a Message Board coordinator role that any robot can switch to. The system would then benefit from the advantages of a centralized and decentralized design.

In order to quantify the performance of the different strategies, future work will look at using repeatable randomized data that can be uploaded into the simulation. We would like to see whether having the central Message Board tailor the list or letting the idle robot do this locally makes a difference to the performance, i.e., the number of help requests that are left unanswered. We also plan to enrich the map terrain and add more scientific instruments/colors (capabilities) to the swarm mix.

Another feature we are interested in implementing involves stopping a robot that is in the process of responding to a help request. This would happen if a robot that is closer to the help request location becomes idle and available. If a robot becomes idle and is closer to a help request location than a robot that is currently en route, the idle robot could take over the task. This would involve sending the idle robot a list that includes help requests that are currently being answered as well as those that are unanswered.

To make the simulation more realistic, we plan to modify the Help Request format to include a time to completion value. This is an estimate created by a robot that has encountered an object or feature that needs to be investigated within a certain time frame due to suspected perishability. In addition, it may be useful to equip the Message Board AM with the ability to detect help requests that have not been answered and where their time to completion is running out.

This paper describes a work in progress, future publications will detail the results of the strategies and compare performance. The goal of the research is to determine if certain collaboration strategies are more suitable to certain situations. We want to measure efficiency of each strategy, metrics will therefore be recorded, these may include: number of steps taken by each swarm robot, number of help requests made, number of help requests that go unanswered. The time taken by each strategy is also important if a foraging scenario is adopted, the time taken to find all items can be compared.

We are currently working on implementing both collaboration strategies into a new simulation where the terrain is identical; this will help us gather statistics on how well they perform in relation to each other. In addition to these strategies we are also implementing a local broadcasting collaboration strategy, which uses nearest neighbour message passing. In a real life scenario this would help solve any signal range communication problems.

Another aspect of our research is to include an Autonomic Overseer Element that can monitor the simulation as it is running, assess the metrics being recorded and then instruct the swarm to change the collaboration strategy it is currently using to one that the Overseer deems more suitable to the terrain or situation. A terrain that is hazardous could result in more swarm casualties, in this sort of situation using a decentralized communication technique that requires a message to be passed via close neighbours (local broadcasting) may be unsuitable. If the swarm is spread too thin then the messages may never reach a sizeable number of robots. The Autonomic Overseer would conclude that the swarm is operating inefficiently and change their mode of communication to a centralized communication strategy. It may choose the Message Board strategy, as this would have the power to send a message to all.

B. Experiments with Physical Hardware

In future we plan to test the cooperation ideas on a small cluster of mobile robots. The simulation approach is enabling us to create swarms with more than one thousand robots; however it would be interesting to compare the simulation results against real life data. For the research we will be using 4 Dr Robot X80-H differential drive style mobile robots. The scenario will mirror the simulations in that the robots will be sent to explore at random. However, for practical purposes, the ‘interesting feature’ or item that they will be searching for will not be as detailed as those used in the simulations. An object or textual sign may be

 ISBN: 978-1-61208-463-3

used to represent the item or terrain feature, necessitating the need to use computer vision algorithms for object detection or OCR in the case of a textual sign.

The system will require some additional features such as the use of wheel odometry information so that each robot can accurately localize itself within the environment. This will allow it to work out where it is and move to another robot's position. The robots will start adjacent to each other in a line; the Message Board will receive movement data from each robot and store this in a global system map. The Message Board's map will enable robots to determine where they are in relation to the rest of the swarm. A similar idea was explored in [15], where a central system stored a map that relied on transmitted robot odometry information. The robot would update cells of the map with positive integers to represent a virtual pheromone trail.

For our experiments, each robot will store a local map and use wheel odometry to work out how far they've moved and if they've changed direction. Actual movements such as 'moved 30cm' will need to be translated into pixel movements and coordinates. If the robot moves 30cm, the simulated version on the local map should also move a set number of pixels. This simulated co-ordinate can then be sent to the Message Board when sending or responding to a help request. The location information within a help request will not be more detailed than the simulated versions. The Message Board will filter the help request list based on proximity; the idle robot will only receive the help requests from robots within a certain distance. The idle robot will need to make navigation decisions locally when deciding how to move to the help request co-ordinates.

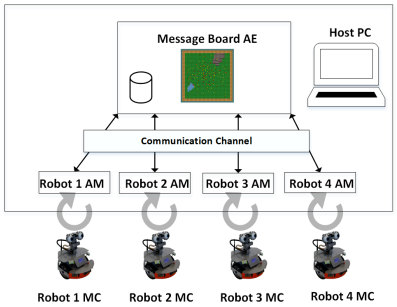


Figure 7. Message Board Autonomic Element and Robot Autonomic Managers running on a PC.

In Figure 7, the system setup is shown, the Message Board Autonomic Element and robot Autonomic Managers all run on the same PC. We may choose to have each running as a separate program and use TCP/IP for interprocess communication; this separation would more accurately mimic a real life scenario. The Message Board AE consists of a Managed Component, which stores all help requests. The MC would also have a virtual Map that could be used to check coordinates and determine proximity when creating a tailored list for an idle robot. Using wheel odometry for localization is flawed and tends to accumulate errors over time. If an idle robot reaches a Help Request location and an odometry error has occurred, it may be necessary to have the robot search the surrounding area for the object.

V. CONCLUSION

This research project aims to apply autonomic computing to the area of swarm collaboration. Instead of replicating the behavior of a natural system, we will create a system were decisions are informed and not reaction based. We believe this type of system would be useful to future space exploration missions where multiple autonomous rovers are sent instead of one all-important rover.

The goal is to simulate multiple direct communication strategies and analyze metrics to determine which strategies perform best in certain situations; a model will then be created from this data. Future work will focus on developing a situational-aware Autonomic Overseer Element that can compare real time data to this model and enable switching between the strategies.

REFERENCES

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

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

[3] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and Autonomic Systems : A Paradigm for Future Space Exploration Missions," IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2006, vol. 36, no. 3, pp. 279–291.

[4] W. Truszkowski, H. Lou Hallock, J. Karlin, J. L. Rash, and M. G. Hinchey, Autonomous and Autonomic Systems with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems. London: Springer, 2009.

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

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

[7] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," Robotica, 2012, vol. 31, no. 3, pp. 1–15.

[8] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella, "Communication assisted navigation in robotic swarms: Self-organization and cooperation," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4981–4988.

[9] C. Christodoulopoulos, C. Kyriakopoulos, and A. Kanatas, "A realistic approach to source localization using a wireless robotic network," The Proceedings of First International Conference on Robot Communication and Coordination, 2007, pp. 1–4.

[10] N. Capodieci and G. Cabri, "Collaboration in Swarm Robotics : a Visual Communication Approach," in 2013 International Conference on Collaboration Technologies and Systems (CTS), 2013, pp. 195–202.

[11] A. Anand, M. Nithya, and T. Sudarshan, "Coordination of mobile robots with master-slave architecture for a service application," in 2014 International Conference on Contemporary Computing and Informatics (IC3I), 2014, pp. 539–543.

[12] C. E. Aguero, V. Matellan, J. M. Canas, and V. M. Gomez, "SWITCH ! Dynamic Roles Exchange Among Cooperative Robots," in Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems - MARS 2006 INSTICC, 2006, pp. 99–105.

[13] M. Veloso and P. Stone, "Video: RoboCup Robot Soccer History 1997 - 2011," in IEEE International Conference on Intelligent Robots and Systems, 2012, pp. 5452–5453.

[14] "Dr Robot X80-H." [Online]. Available: http://www.drrobot.com/products_item.asp?itemNumber=X80-H. [Accessed: 04-January-2016].

[15] I. Susnea, A. Filipescu, V. Minzu, and G. Vasiliu, "Virtual Pheromones and Neural Networks Based Wheeled Mobile Robot Control," in 13th WSEAS International Conference on Systems, 2009, pp. 511–516.

 ISBN: 978-1-61208-463-3

Autonomic Self-Adaptive Robot Wheel Alignment

Martin Doran, Roy Sterritt, George Wilkie
Faculty of Mathematics and Computing
University of Ulster
Jordanstown, N.Ireland
doran-M18@email.ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk

***Abstract*—The Autonomic Computing paradigm was first presented almost 15 years ago as a 20-30 year long research agenda. Organizations like NASA have explored the possibilities of an autonomic system along with the biologically inspired Swarm and Agents approaches. These Systems are unfolding circumstances rather than preconceived scenarios. This paper focuses on the aspect of the Autonomic System, were, adaptive self-optimization are explored using an intelligent machine model layers such as Reaction, Routine and Reflection. In the experimentation, a Pioneer P3-DX Robot is used to simulate a Planetary Rover. The Pioneer Robot has been the subject of hardware faulting – in the case Wheel Integrity. Using the Microsoft Robotics Developer Studio (MRDS) and Microsoft SQL Server, a framework has been developed, that records the Pioneer Robot's sensor data before and after hardware faulting. The Autonomic system elements, such as self-adaptive and self-optimizing are used to process the data. The Reaction, Routine and Reflection responses are determined at the appropriate response rates from the evaluation of these and form part of the self-healing, self-configuring, self-optimizing and self-protecting strategy to enable the Robot to continue to function, even with hardware issues such as a faulty wheel.**

Keywords—autonomic, self-adaptive, self-optimizing

I. INTRODUCTION

The future of space exploration will most certainly involve the deployment of vehicles such as planetary Rovers. Although Rovers like Curiosity and Opportunity have been successful, they have a limit in mobility and surface coverage. NASA in the past had researched the idea of Swarm Agents, such as multiple Rover deployments [1]. Recently, NASA have begun testing with "Swarmies" [2], were multiple Rovers can be deployed to gather resources on Moons or on Planets. Other scientific research has put forward the idea of "honeybee search strategy" [3], were large quantities of robots can been deployed to gather important data in special areas of interest. Fundamentally, the design of each individual Rover would be relatively basic to keep costs at a minimum. If each Rover is functioning under an Autonomic Management System, then the faults that occurred would be managed to the extent that the Rover could still function in its mission goals. Current NASA missions have reported hardware faults. The Curiosity Mission reported faults on all six wheels on the Rover; were the rubber casing on each wheel had been punctured through by sharp rock material [4]. Consequently, Mission Control was forced to plan alternate routes to avoid certain types of rocky outcrops. This ultimately has an impact on mission time and mission objectives. Could the wheel issue with Curiosity been identified earlier if the Rover had an Autonomic Self-protecting System [5] onboard? This paper investigates the implementation an Autonomic System and how it can deal with hardware failures.

Using a mobile robot to simulate planetary Rovers, we exposed the robot to hardware failure such as a wheel fault. At first, the robot is tested using two wheels in perfect order. The robot is given a task to travel a given distance, from a start point and then onto an end point. Each journey is recorded in a SQL database, using the robot sensor data. The robot is then fitted with one slightly damaged wheel. The robot is then given the same task as previous explained, with the results recorded in a SQL database. This paper focuses on implementing an Autonomic System to monitor and analyze the data, plan any necessary changes and execute those changes [6].

The structure of this paper is as follows. Section II documents related work in autonomic detection of sensor faults. Section III documents the autonomic system architecture. Section IV documents the autonomic management system and framework architecture. Section V documents the robotic hardware used in the research. Section VI documents wheel damage scenarios. Section VII documents autonomic failure detection in robotic mobile hardware. Section VIII documents the software framework and state machine used to create tasks including robot motion, laser readings and database recording. Section IX documents processing of the data collected from the tasks performed by the robot. Section X documents the equation used to compensate for the robot wheel alignment error. Section XI concludes the paper and outlines future work.

II. RELATED WORK

Since the introduction of autonomic computing [6], there have been a number of approaches on the subject of hardware fault-detection in mobile robots. Loss of sensor data is a typical example. The software framework in a mobile robot can be setup to ask for sensor information regardless of the sensor device that supplies it. If an *interface* program requires an object-detection *service*, then the "distance" value can be acquired from say, a laser range-finding sensor; however, if the laser sensor would fail, then the *service* can switch to an ultra-sonar sensor to obtain the same "distance value" [20].

The study of the types of failure that occur in mobile robots, libraries or classes of autonomic properties could be

 ISBN: 978-1-61208-463-3

developed that address each type of failure. Services can be provided that correspond to the type of failure found, such as service oriented architecture or a multi-agent system [21].

Further work can be seen in the use of Distributed Integrated Affect Reflection Cognition Architecture (DIARC). DIARC is knowledge-based architecture that can employ human-robot interactions without any structural modifications. DIARC employs MAS (multiple agent systems), which allows the distribution of components over multiple hosts and thus providing support for the autonomic detection of component faults and for subsequent error recovery [22].

Other fields of work such as those found in Organic Computing (OC), have experimented with similar attributes found in autonomic computing such as *self-adapting* and *self-healing*. Experiments using *hexapod* robots show that even with the amputation of one of the legs, the robot can still function by re-configuring the neighboring legs to compensate for the missing leg. [24].

The Related Work contributions all involve the detection of hardware failure using autonomic principles. However, they make no argument for trying to compensate for the error detected by employing specialized algorithms that will make use of the affected hardware, even if the *sensor* or *effector* operational ability is greatly reduced.

III. AUTONOMIC SYSTEM

The autonomic system can be summarised by four objectives: self-configuring, self-healing, self-optimizing and self-protecting; additionally, with four attributes: self-awareness, self-situated, self-monitoring and self-adjusting [7]. Self-configuring has the ability to automatically make adjustments when faced with changing circumstances. Self-healing is concerned with dealing with unexpected faults. It can recover from these faults and where possible, repair the faults. Self-optimizing has the knowledge of expected performance values. It can use policies to maintain optimum performance but also flexible to employ new policies to enhance performance. Self-protecting can deal with external attacks. It can establish what could potentially be a threat and how to deal with those threats. The autonomic manager describes the Monitor Analyse Plan Execute (MAPE) loop. This 'loop' is connected to a *knowledge* block [6]. This connection indicates that knowledge is used throughout the autonomic manager.

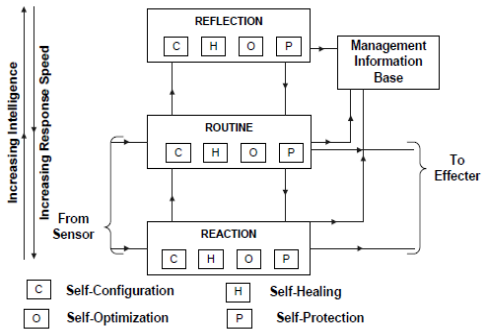


Figure 1. IMD Architecture.

The Intelligent Machine Design (IMD) architecture provides the distinct layers (see Figure 1): Reaction, Routine and Reflection [8].

The Reaction Layer is connected to effectors and sensors. This is the lowest level were no learning occurs. The Routine Layer will handle know situations. Input is received from both the Reaction and Reflection layers. The Reflection Layer makes decisions based on knowledge collected over a period of time. Operations at this level are executed based on experiences, current behavior and current environment. The lowest intelligence is found in the Reaction Layer, whereas important decision making (higher intelligence), is found in the Reflection Layer.

IV. AUTONOMIC ANALYSIS

In this section, we look at how the combination of the MAPE-K Loop [6] and the IMD Reflection [8], can be used to design an Autonomic Management System (see Figure 2).

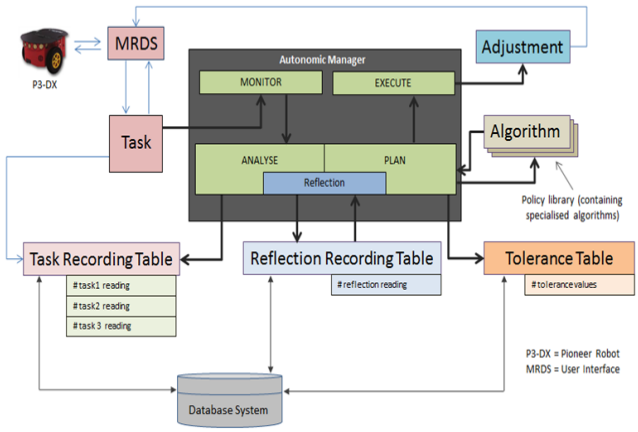


Figure 2. Autonomic Management System.

Figure 2 shows how the Pioneer P3-DX robot is controlled using the MRDS interface [15] (detailed in Section VIII). To emulate a robot mission, the P3-DX robot is set a number of tasks. The data relating to these tasks is recorded into a Database Management System. The tasks are closely monitored by the *MAPE-k* Monitor component; monitoring information can then be passed onto the Analyze function. The *MAPE-k* Analyze component can then use the data supplied by the *Reflection Layer* to evaluate if there is any anomalous behavior in the P3-DX *task* data. The *MAPE-k* Plan component will then use the data supplied by the *Reflection Layer*, to decide what *algorithmic policy* is required to compensate for the anomaly discovered. The *MAPE-k* Execute component will initiate the *policy* into the program *loop* - these *adjustment* values are then passed onto the MRDS to instruct the P3-DX robot command functions.

The *knowledge* contained into the database system, forms an integral part of the autonomic management system. The *Reflection System* maintains a model of self-representation. Reflection enables inspection and adjustment of a system at

 ISBN: 978-1-61208-463-3

run-time [19]. The P3-DX robot tasks history is recorded into database *tables*; analyzing this history, allows the autonomic management system to make *self-adjustments* based on hardware sensor performance of the robot.

Solutions for traditional fault tolerance systems are usually designed and configured at design time. The Developer is tasked with identifying in advance the most critical components and then, decides what strategies to use to overcome possible faults [18]. Autonomic Systems are designed to look for subtle changes in behavior or inconsistent performance data. The autonomic element has its own manager system. The managed system looks after the controller. The controller consists of two loops – the local loop and the global loop [17]. The global loop will run constantly, gathering data from sensors and storing this data in the database system. The global loop manages the behavior of the whole system. The local loop is *blind* to the overall system loop. The local loop will focus on analyzing data in the database and look for discrepancies. Discrepancies can be identified by comparing the data with known tolerance values. If the tolerances are above the limits that the local loop can maintain, then this can affect the performance of the overall system. This change in performance will trigger the global loop. The global loop will then implement a policy to deal, in this case, with the wheel alignment issue.

V. PIONEER P3-DX MOBILE ROBOT

The Pioneer P3-DX is a mobile robot with two independent drive wheels, plus an additional caster for stability (see Figure 3). The internal drive uses Proportional-Integrated Derivative (PID) system with a wheel encoder feedback to adjust a pulse-width-modulation (PWD) at the motor *drivers* to control the power of the motors [9]. The P3-DX is also fitted with a LMS 200 Laser. The LMS 200 is capable of measuring out to 80m over 180° arc. The sensor operates by shinning a laser of a rotating mirror. As the mirror spins, the laser scans 180°, effectively creating a fan of laser light (see Figure 4). Any object that breaks this fan reflects laser light back to the sensor. The distance is calculated based on how long the laser takes to bounce back to the sensor [10].

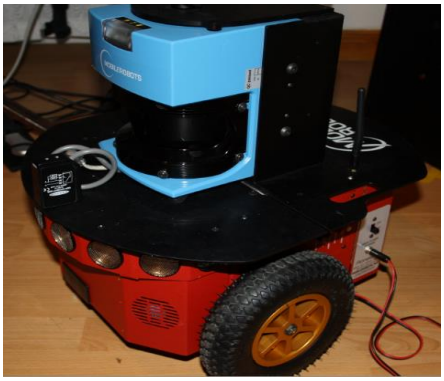


Figure 3. Pioneer P3DX mobile robot fitted with LMS 200 laser.

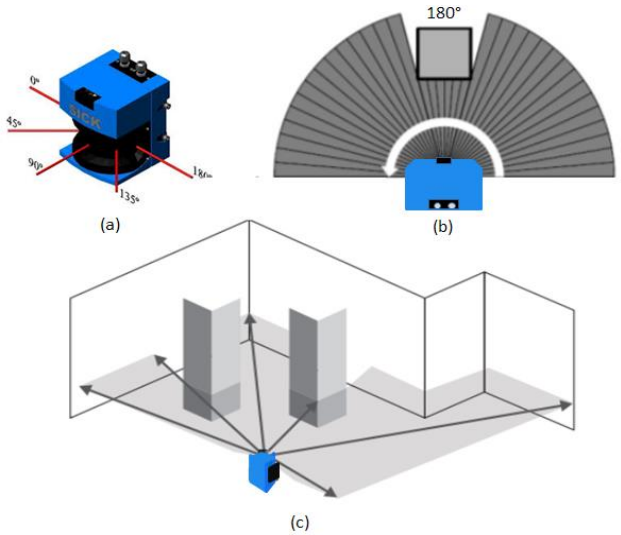


Figure 4. (a) The LMS 200 laser has a 0° to 180° field of view. (b) The laser creates a fan of laser light that scans from right to left. (c) Objects are detected by breaking the laser fan projection. The distance is detected by the time it takes for the laser to bounce back to the sensor.

The Pioneer P3-DX robot is used to simulate a planetary rover. The robot has an on-board PC and a WIFI connection. The software framework used to control the robot is MRDS.

VI. WHEEL DAMAGE

One of the most important aspects of a planetary rover mission is its ability of movement on the surface. The rover is reliant on optimal wheel performance, in-order to reach locations according to the mission goals. Possible wheel damage to a rover on a mission is very difficult to anticipate. In our own everyday lives, wheel damage can occur in with our cars, motorbikes and pedal bikes – hitting objects on the road or potholes, causes the most damage. The possibility of damage to a wheel on a rover mission is very high, as we have witness with the Curiosity Mission (see Figure 5) [13].

Figure 5. Curiosity wheel damage – many cracks like this have been found on all six wheels of the rover [13].

NASA engineers are looking to provide a software update to Curiosity rover; the software could provide the ability to

match electrical current with wheel drive [14]. This type of damage can lead to issues such as wheel alignment. If the wheel can't drive in a straight line as expected, then there would need to be a *self-adjusting* process employed to allow the mission to continue.

VII. AUTONOMIC FAILURE DETECTION

Physical failures in a planetary rover can affect systems, such as effectors, sensors, power and communication [11]. In this paper, we investigated the failure in the effector systems centering on wheel degradation. The investigation was begun by fitting two wheels in perfect working order, onto the Pioneer P3-DX Robot. The robot is then given a series of tasks where it is required to travel from known start-point and then move to a known end-point. For tests purposes, the route that the robot is traveling along is parallel to the laboratory wall. The test consisted of the robot being positioned at the start-point exactly parallel to the wall at a given distance. The onboard LMS 200 laser is used to accurately record the distance from the robot to the wall. The robot is then given a command to move a given distance. When the robot arrived at the destination, the LMS 200 laser is used to record the distance from the robot to the wall. The results of the laser data were recorded into an SQL Server database. These tests were repeated multiple times to give us an accurate evaluation of the robot performance with two perfectly operational wheels. Standard Deviation equation is applied to the test results to give an indication of the accuracy of the robot's movement from a start-point to an end-point. The Standard Deviation results were then evaluated using an algorithm to represent the Autonomic Intelligent Machine model Reflection layer. This *reflection algorithm* processed the data to establish if the robot wheel alignment was accurate against expected tolerance values. This self-monitoring of the robot over a given period of time, confirms that the robot is operating as expected. The Pioneer Robot is then fitted with a slightly damaged wheel. The robot is then evaluated using the same test process. The purpose of this action was to process the data within the *reflection algorithm*, and initiate the Autonomic *self-monitoring* to cause the system to identify there was a problem and consequently put in motion, policies that could repair the issue or at least compensate for the error. The evaluation done in the Reflection layer would subsequently initiate a policy within the Routine layer, which in turn would cause a physical implementation in the Reaction layer.

VIII. SOFTWARE FRAMEWORK

Software Development for this paper is carried using the MRDS framework. MRDS is a service-oriented programming model that allows the creation of asynchronous and state-driven applications [15] and [16]. Code development is carried out using C# language. Database work was completed using Microsoft SQL Server and User defined stored procedures.

To create robot tasks (explained in Section VII), it required implementation of an event driven and state-based behavior processing, using a state machine as shown in Figure 6. Between each state, transitions are added. These transitions are triggered by notifications received from partner services [12].

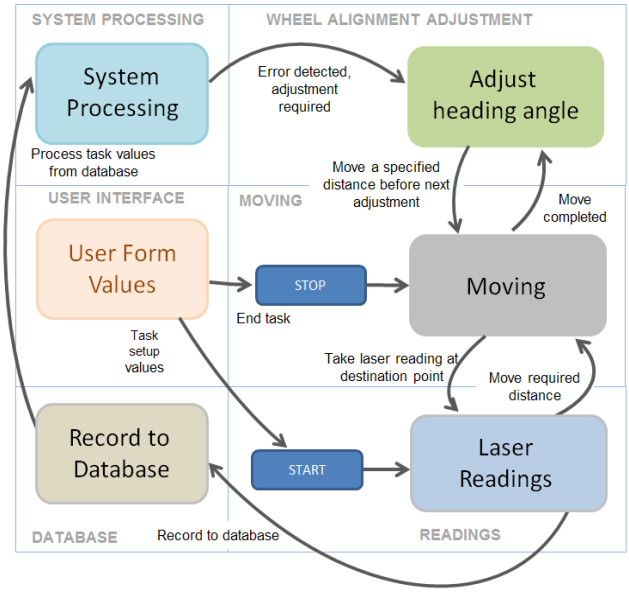


Figure 6. Shows the state machine used to create the robot tasks, process database information and make necessary adjustments if an error is detected.

The System Processing state is executed after the robot has completed a task; this *self-monitoring* attribute can then initiate a *self-adjusting* process if an error is detected.

IX. DATA EVALUATION

The graphs in Figure 7 shows the readings taken from the robot tasks (see Section VII). Graph (a) shows how the robot performs with both wheels fully functional. The robot stays within the limits of the *expected* path. Graph (b) shows how a damaged wheel causes the robot to veer off to the right.

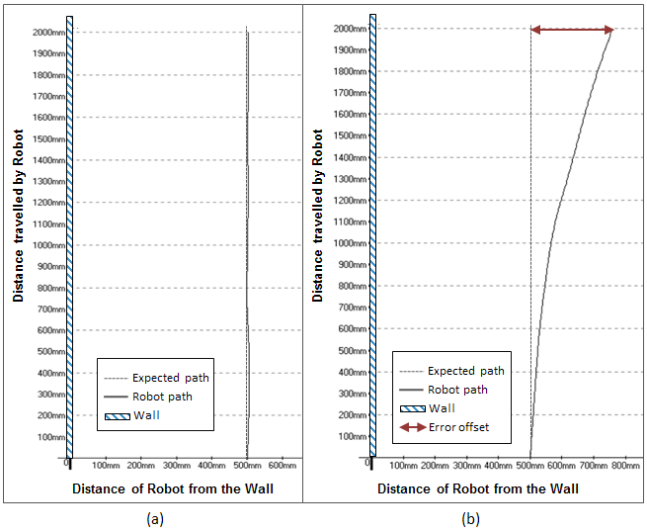


Figure 7. Graph (a) shows the path of the robot with both wheels at optimal performance. Graph (b) shows the path of the robot with one wheel in a damaged state.

 ISBN: 978-1-61208-463-3

Using Standard Deviation equation (non-grouped data), the results from two types of tests (explained in Section VII) were calculated.

$$\sigma = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} \quad (1)$$

The first sets of tests are conducted using the Pioneer P3DX robot with two wheels in perfect condition – Test Scenario A. The second sets of tests are conducted using one wheel with slight damage – Test Scenario B. Table I shows the standard deviation value for Test Scenario A and how adding the results of Test Scenario B affect the standard deviation value.

TABLE I. TEST VALUES FOR WHEEL ALIGNMENT CALCULATIONS

Pioneer P3DX wheel alignment testing - the numbers represent the amount in millimeters (mm) that the robot was from its required destination point, after each task.										
Test Scenario A: for a Robot with two wheels in optimal condition										***SD***
2	9	-9	4	-5	-7	-22	-7	-6	-5	8.63
-8	-13	-13	11	-14	-4	-13	-2	-6	-6	
3	-21	-12	-10	4	-10	9	-11	-4	-21	
-5	10	-8	2	-7	-12	3	-5	-14	10	
6	-2	-7	4	-13	5	8	3	-4	7	
Adding another one test result to Scenario A, does not affect the SD value significantly										***SD***
									-5	8.54
Test Scenario B: for a Robot with one slightly damaged wheel, the overall SD changes significantly and thus would be flagged as a fault										***SD***
								35	49	12.02

Significant changes to the standard deviation (SD) value shown in Table I indicated that there was a problem with wheel alignment in the Pioneer P3-DX robot. The state machine System Processing discussed in section VIII was implemented to identify changes in SD values. Identifying the error is achieved by comparing the SD value to the set *tolerance* value. If the SD is within the tolerance value range, then no action is needed; if the SD is above the tolerance value, then an *error* task was executed. This then resulted in the robot initiating a *self-adjustment*, were an algorithm is used to determine the *degree* of the error and calculate the values needed to compensate for that error.

X. WHEEL ALIGNMENT ERROR EVALUATION

To compensate for the wheel alignment fault, an algorithm is required to work out the compensation value needed to correct the robot trajectory. The aim of this process is to keep the robot functioning even with a damaged wheel. As the robot moves, the damaged wheel is constantly pulling it away from its expected path. To correct the wheel alignment error, the robot needs to adjust its heading at calculated intervals during its journey. To achieve this, the robot would travel a certain distance, stop, then, turn itself back toward its expected path, then, move again. The downside of this particular strategy is that it will add significant distance and time to the robot mission. However, this is justified, in that the robot will arrive at its expected destination point rather than being significantly off-course.

Using the values from Test Scenario B, an *average* distance from which the robot was from its expected destination is calculated. Using Right-Angled Triangle equation, the angle between the *Hypotenuse* side and the *Opposite* is calculated, see Figure 8.

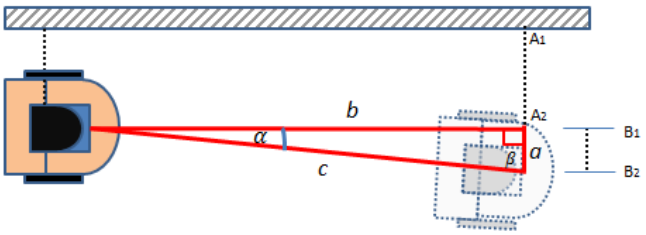
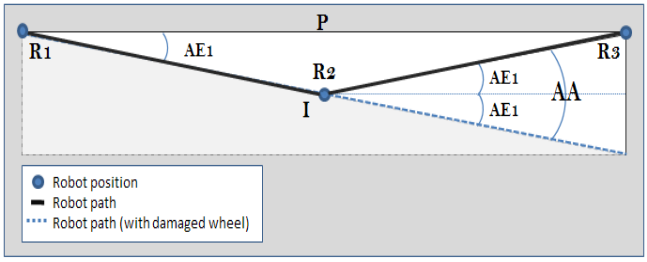


Figure 8. The Pioneer P3DX robot with a damaged wheel: this caused to the robot to slew to the right. A1 to A2 represents the expected distance the robot should be from the wall. B1 to B2 represents the average distance the robot was offset from the expected destination point.

$$\alpha = sin^{-1}\left(\frac{a}{c}\right) \quad (2)$$

The equation (2) is used to calculate the *angle* of the initial error-offset discovered from results in Figure 7 (b). This *angle error* value is then used to establish the *angle of turn* needed for the robot to make its heading adjustment; see Figure 9.



(a) Robot error correction using 1 interval

Figure 9. Represents how the *angle of turn* is calculated.

The *angle of turn* calculation is formulated using the values represented in Figure 9. The 'R', represents the robot position. When the robot reaches the 'I' position (*interval*), the robot is stopped. The robot *heading angle* is then adjusted and the robot continues its journey. The 'AE' represents the *angle* of the wheel alignment error calculated using equation (2). The 'AE' angle value is then *doubled*. The reasoning behind this is that two 'AE' values are required to bring the robot back to the expected path. The two 'AE' values are then divided by the number of *intervals* the robot is required to stop. The 'AA' represents the *angle*

of turn needed to allow the robot to re-establish the expected journey path marked as 'P'.

$$AA = \frac{2AE}{I} \quad (3)$$

From the values represented in Figure 9, we can derive the equation (3). The *interval* 'I' represents the number of times the robot with stop and adjust its heading angle. The more *intervals* the robot uses, the more accurate the robot will be in terms of keeping to the original journey path. In equation (4), *interval distance* is represented by 'ID' and *total distance* is represented by 'TD'. The *interval distance* is calculated as follows:

$$ID = \frac{TD}{I} \quad (4)$$

Figure 10 shows how the number of intervals used decreases the *error-offset* value. Table II shows a robot with wheel damage, driven over a fixed distance. The robot is stopped and adjusted according to the number of intervals applied. The offset value is the maximum distance the robot is from the expected path.

TABLE II. COMPARE OFFSET VALUES USING A GIVEN INTERVAL #

Pioneer P3-DX wheel alignment testing. Error offset decreases as the number of intervals increases. The maximum offset is measured from the expected path value.			
Distance of Journey	Number of Intervals	Angle of adjustment	Maximum offset error value
2000 mm	1	12°	44 mm
2000 mm	2	6°	26 mm

The graphs in Figure 10 show the comparison of the robot path when used with different *interval* values.

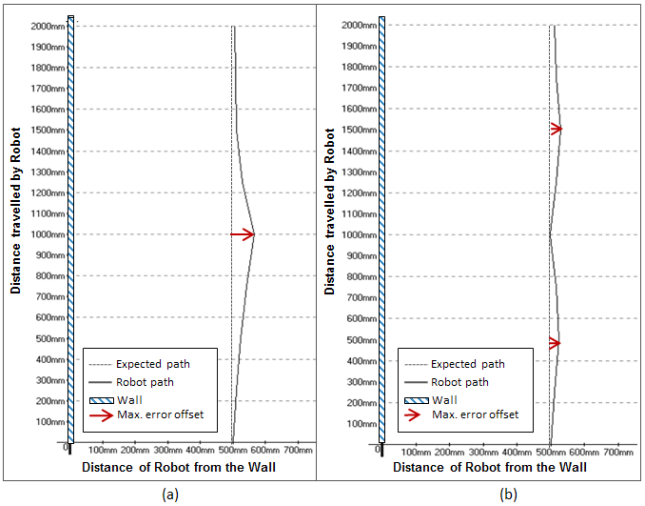


Figure 10. Using the compensation algorithum, the robot journey accuracy is increased when the number of intervals is also increased. (a) Robot journey uses one interval. (b) Robot journey uses two intervals.

This compensation method reflects the ability of the robot to *self-adjust* itself to arrive at the expected destination point even with a damaged wheel. Figure 11 shows how the robots journey is divided into *intervals*. This process is repeated until the robot reaches its destination point.

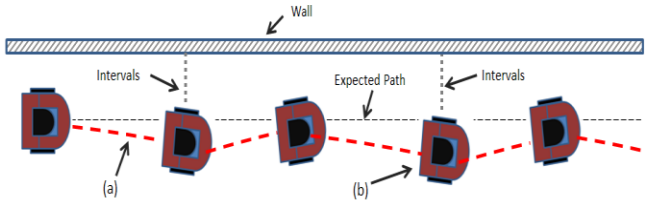


Figure 11. Shows the path of the Pioneer P3-DX robot when it implements the compensation algorithm. (a) Wheel damage causes the robot to 'slew' away from the expected path. (b) When the robot reaches an interval point, the robot is stopped and then turned on its axis at the required angle for adjustment..

The severity of the alignment error will have an effect on the 'actual' distance and journey time of the robot; as the *angle* of compensation increases, the journey time also increases. On a shorter journey, this may not be a factor but if the robot needs to travel for long distances, then could have an impact on resources like power consumption.

Algorithm 1 Wheel Alignment Compensation

```
1:  Read test data from SQL Tables.
2:  for (ts = number of tests) do
3:    for (vs = each value in test[ts]) do
4:      Calculate the average value (av) of each test vs[ts]
5:      Use SD equation to test if the av > tolerance range
6:    if ( tolerance range is exceeded == true)
7:       Alignment Error Angle (ae) = sinθ (Right-Angle) equation
8:       Angle of Adjustment (aa) = 2 * ae/number of intervals
9:    end if
10:   Enter the journey values for the Robot.
11:   dis = distance to travel
12:   ni =number of required intervals
13:   aa= alignment error angle
14:   cd = current Robot distance
15    iv (Interval value) = dis/ni
16:   iv = interval value
17:   while cd< dis do
18:   Adjust the Robot direction at interval setting (iv)
19:   if (dis % iv == 0)
20:      StopRobot()
21:      Rotate robot to new angle direction(aa)
22       RotateRobot(aa)
23:      MoveRobot()
24:   end if
25:   end while
```

Figure 12. Shows pesudo code for the Robot Wheel Alignment compensation.

Figure 12 shows a representation in pseudo code for the Robot wheel alignment data processing and compensation functions. The initial SQL data from the robot tasks is processed and checked against know tolerance values. If the tolerance values are exceeded, then the *compensation algorithm* is employed to re-establish the robot to its expected journey path.

 ISBN: 978-1-61208-463-3

TABLE III. TEST VALUES WITH WHEEL COMENSATION ALGORITHM

Pioneer P3-DX wheel alignment testing - the numbers represent the amount in millimeters (mm) that the robot is from its required destination point, after each task.										
Test Scenario C: for a Robot with damaged wheel and using compensation algorithm										*SD*
2	9	-9	4	-5	-7	-22	-7	-6	-5	8.63

Table III shows how the *compensation algorithm* restored the robots wheel alignment measurement (SD), within the expected tolerance values.

XI. CONCLUSION AND FUTURE WORK

The purpose of this research paper is to identify how the autonomic model can be applied to dealing with robot hardware issues, such as *wheel alignment*. If NASA decides to deploy *swarm* planetary rovers in the future, then autonomic systems will need to be seriously considered, to deal with possible hardware degradation and software system failures.

The Intelligent Machine Design theory proves that responses such as Reflection can over time, analyze data and predict possible issues at an earlier stage. This can be put to the test by carrying out more extensive testing in the future and building up a knowledge database. The *compensation algorithm* we applied in these test cases is only one of many ways which the robot wheel alignment issues could have been solved. In the future, we would like to investigate the use of *wheel velocity variation* for dealing with wheel alignment issues; were small increments of power can be applied to a damaged wheel. Unfortunately at present, such robots like the Pioneer P3-DX do not possess the *fine granularity* in wheel velocity, which is required for such an experiment.

REFERENCES

[1] R. Luna, A. Oyama, and K. Bekris, "Network-Guilded Multi-Robot Path Planning for Resource-Constrained Planetary Rovers," IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'10) on, September 2010, pp. 776-783.

[2] National Aeronautics and Space Administration. NASA: Meet The Swarmies-Robotics Answer To Bugs. [Online]. Available from: https://www.nasa.gov/content/meet-the-swarmies-robotics-answer-to-bugs/2016.02.02

[3] Á. Kisdi and A. R. L. Tatnall., "Future robotic exploration using honeybee search strategy: Example search for caves on Mars," Acta Astronautical , vol 68, issues 11-12, pp. 1790-1799, June-July 2011.

[4] National Aeronautics and Space Administration. NASA: How Wheel Damage Affects Mars Rover Curiosity's Mission. [Online]. Available from: http://www.space.com/26472-mars-rover-curiosity-wheel-damage.html/2016.02.02

[5] R. Sterritt and M. G. Hinchey, "Engineering ultimate self-protection in autonomic agents for space exploration missions, "Engineering of Computer-Based Systems," 2005. ECBS '05. 12th IEEE International Conference and Workshops on, April 2005, pp. 506-511.

[6] D. M. Chess, A. Segal, I. Whalley, and S. R. White, "An architectural blueprint for autonomic computing," IBM Corporation, 2004.

[7] R. Sterritt and M. G. Hinchey, "Biologically-inspired concepts for self-management of complexity," Engineering of Complex Computer Systems, 2006. ICECCS 2006. 11th IEEE International Conference on , August 2016, pp. 65-70.

[8] H. Shualib, R. J. Anthony, and M. Pelc, "Framework for Certifying Autonomic Computing Systems," The Seventh international Conference on Autonomic and Autonomous Systems on, January 2011, pp. 122-127.

[9] Adept Mobile Robots. Pioneer 3 Operations Manual, Version 6, 2010.

[10] J. Bedkowski, M. Kretkiewicz, and P. Maslowski, "3D Laser Range Finder based on rotated LMS SICK 200," unpublished.

[11] E. Matson and S. Deloach, "Integrating Robotic Sensor and Effector Capabilities with Multi-agent Organizations," Proceedings of the 2004 International Conference of Artifical Intelligence (IC-AI-04) on, June 2004, pp. 481-488

[12] J. S. Cepeda, L. Chaimowicz, and R. Soto, "Exploring Microsoft Robotics Studio as a Mechanism for Service-Oriented Robotics," (LARS) on, October 2010, pp. 7-12.

[13] NASA/JPL-Caltech/MSSS; Sol 962, April 21, 2015.

[14] National Aeronautics and Space Administration. NASA: Mars Rover Curiosity Dealing With Damage. [Online]. Available from: http://www.space.com/29844-mars-rover-curiosity-wheel-damage.html/ 2016/02/04

[15] Microsoft. Microsoft Robotics Developer Studio. [Online]. Available from: http://www.microsoft.com/robotics/2016/02/04

[16] K. Johns and T. Taylor, Professional Microsoft Developer Studio, Wiley Publishing Inc., 2008.

[17] M. Parashar and S. Hariri, "Autonomic Computing: An Overview," International Workshop UPP on, September 2004, pp. 257-269.

[18] R. Almeida, J. Briot, S. Aknine, Z. Guessoum, and O. Marin, "Towards Autonomic Fault-Tolerance Multi-Agent Systems," The 2nd Latin American Autonomic Computing Symposium (LAACS'2007), Petropolis, Rio De Jeniro, Brazil, September, 2007.

[19] P. Maes, "Computational Reflection," The Knowledge Engineering Review on, November 1988, pp.1-19.

[20] N. A. Melchior and W. D. Smart, "Autonomic systems for mobile robots," in Autonomic Computing, 2004. Proceedings. International Conference on, May 2004, pp. 280-281.

[21] C. Rouff, J. Rash, and W. Truszkowski, "Overcoming Robotic Failures through Autonomicity," in Engineering of Autonomic and Autonomous Systems, 2007. EASe '07. Fourth IEEE International Workshop on, March 2007, pp. 154-162.

[22] M. Scheutz and J. Kramer, "Reflection and Reasoning Mechanisms for Failure Detection and Recovery in a Distributed Robotic Architecture for Complex Robots," in Robotics and Automation, 2007 IEEE International Conference on, April 2007, pp. 3699-3704.

 ISBN: 978-1-61208-463-3

Adaptive Construction Behavior in Robot Swarms

Yara Khaluf

Department of Information Technology
Ghent University
Ghent, Belgium
Email: yara.khaluf@ugent.be

***Abstract*—In this paper, we investigate the collective behavior of a robot swarm that emerges in constructing walls for isolation purposes. Collective construction is one of the highly required behaviors for the future applications at which robotics systems are planned to be deployed. The construction task is performed using a swarm of homogenous robots. We, furthermore, present a probabilistic approach that allows us to design an adaptive construction behavior. Our results are verified using physics-based simulations.**

Keywords–Robot swarms; Collective construction; Adaptive behavior.

I. INTRODUCTION

One of the fascinating self-organized behaviors in nature is collective construction, that is observed by many social insects such as ants and bees [1]. Collective construction refers to the ability of simple individuals to achieve the construction of complex structures in a self-organized manner by following a set of simple rules. A prominent example can be found by termites, which are able to build complex and huge mounds without any detailed plan. Termites live in societies where the collective power outstrips that of the individuals. They communicate between each other directly and indirectly through their environment (Stigmergy) in order to take decisions related to depositing their pieces of building material, see [2]. Swarm robotics is a promising approach in which a large number of simple robots collaborate to achieve a goal beyond the capability of an individual robot, see [3]. Swarm robotics was mainly inspired by natural swarms and has inherited their advantages including fault-tolerance, scalability, and flexibility. Those advantages allow swarm robotics to provide an efficient solution for a wide range of applications, in which constructing particular structures may be a fundamental task.

In this paper, swarm robotics is used to achieve a collective behavior that results in constructing a wall to separate a working arena in two parts. This task can be used in real scenarios to prevent the access from one part of a particular arena to its another part or to isolate dangerous parts. Construction tasks can be an important part of military tasks, agriculture tasks, civil tasks, and others. We start by designing a self-organized construction behavior. Afterwards we present a probabilistic approach in order to turn our behavior into an adaptive one that can deal with the various dynamics of the environment, e.g. recognizing the construction progress and thus the end of a construction task or the need to isolate a new part of the arena by constructing a new wall. The rest of the paper is organized as in the following: Section II introduces the wall construction problem and presents the performance metrics (quality metrics) used in measuring the quality of the obtained solutions. Section III is dedicated to discuss the related work that has focused on constructive behaviors in natural and robot swarms. The construction behavior of a robot swarm is described in Section IV, in which both the general and the adaptive construction behaviors are presented. A set of physics-based simulations and the discussion of their results are presented in Section V and the paper is concluded in Section VI.

II. PROBLEM DESCRIPTION

A homogeneous swarm of N robots – all are foot-bots having the same hardware – is used to build a wall that isolates an undesired (dangerous) part of the arena from the rest of the arena. The wall should be constructed in a way that access is prevented between the two arena parts. The location at which the wall is required to be constructed is referred to as the construction area, which can be indicated using specific environmental parameters. In our scenario, the construction area is indicated by using a black strip on the floor. Furthermore, we assume that M building blocks (cylinder shaped) are scattered initially at the safe side of the arena, where M is equal to or greater than the amount of blocks required to build the desired wall. The building blocks are identical and the block can be transported using a single robot. The required height of the wall is equal to the height of a building block. Thus, the wall is built using a single layer of blocks i.e., no vertical building is needed. Since no exact notions of the building process are encoded in the robots' behavior, constructing the required wall represents a serious challenge.

Our goal, as mentioned above, is to isolate a dangerous part of the arena from a safe one. Therefore, the constructed wall should prevent intrusions by being as compact as possible. Moreover, it should provide the maximum coverage possible over the boarder between the two sections of the arena, and finally, it should consume as less building blocks as possible to preserve such blocks for potential construction tasks in the future. We define three performance metrics to measure the quality of the wall that needs to be constructed along the y-axis of the arena, see Figure 1 for illustration:

- **The wall thinness** is defined as:

$$W_{thinness} = \begin{cases} 0 & \text{if } M = 0, \\ 1 - \dfrac{\sigma(X)}{x_{max} - x_{min}} & \text{otherwise} \end{cases}$$

  where σ is the standard deviation and X is the set of the x coordinates of the building blocks within the construction. x_{max} and x_{min} are the maximum of the

 ISBN: 978-1-61208-463-3

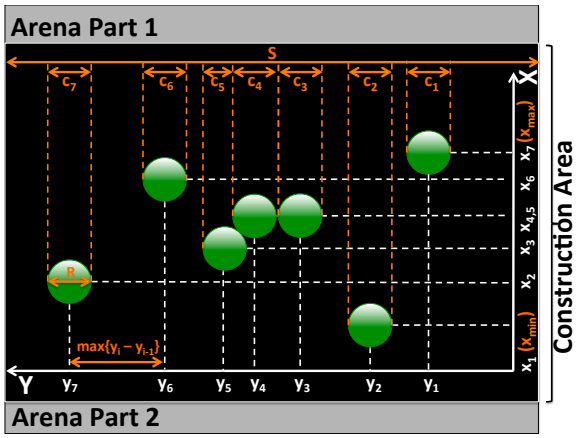


Figure 1. Illustration of the quality measures used in the performance metrics.

x coordinates and the minimum of the x coordinates, respectively. The thinness measure takes its values in the range $[0, 1]$ and an optimal solution is the one which maximizes this measure and thus leads to build a thiner wall.

- **The wall compactness** is defined as:

$$W_{compactness} = \begin{cases} 0 & \text{if } M < 2, \\ \dfrac{max\{y_i - y_{i-1}\}}{R} & \text{otherwise} \end{cases}$$

Where $max(y_i - y_{i-1})$ is, as illustrated in Figure 1, the maximum distance between two consecutive blocks on the y-axis. R is the diameter of a construction block. Compactness is best when it is close to 1. A value lower than 1 means that the maximum distance (projected onto the y-axis) between the center of the blocks is smaller than R. Whereas, a value larger than 1 means that the maximum distance between the center of the blocks is smaller than R.

- **The wall coverage** is defined as:

$$W_{coverage} = \begin{cases} 0 & \text{if } M = 0, \\ \dfrac{\sum_{i=1}^{\#placed\ blocks} c_i}{S} & \text{otherwise} \end{cases}$$

Where c_i is the diameter projection of the i-th block in the construction area on the y-axis (in the sum we remove the overlaps of the projections) as shown in Figure 1. S is the size of the construction area along the y-axis. Coverage takes values in $[0, 1]$ and an optimal solution is the one which maximizes this measure and thus leads to a larger coverage.

III. Related Work

Collective construction is a well-known behavior in natural swarms such as social insects, in which complex structures are achieved in a self-organized manner. Authors in [4] have reviewed some of the basic mechanisms used by social insects to build structures such as mounds by termites, brood structures in honey bees, and others. Other examples can be found in [5] and [6]. Furthermore, several authors tried to formalize the construction behaviors observed in nature in models, such as in [7] and [8].

In swarm robotics and based on its significant importance, collective construction was tackled in several works. In [9], the authors proposed a system of 20 small bulldozers employing simple rules to level the ground at a lunar construction site. In [10], the authors have introduced a minimalist solution inspired from the behavior of the Leptothorax tuberointerruptus ants to build a defensive wall using two templates which were created once using a white strip and second using halogen lights. The presented approach is relatively inflexible where turning angles and particular traveling distances are hard coded. The authors in [11] have tackled the problem of constructing a wall in which blocks should be of alternated colors and robots communicate to agree on the next feasible color of the building block. This work focuses on the role the communication plays in coordinating the behaviors of the robots. This is illustrated by showing that exchanging the color of the last block reduces the attempts to place a block of the same color as next. Authors in [12] have developed a mathematical model based on a particular species of ant called blind bulldozing, for the purpose of clearing an open area out of rocks. In [7], the authors considered construction problem without addressing the issue of generating pre-specified structures. Some works have presented approaches, in which the building blocks are the mobile robots themselves as in [13] and [14]. In these approaches, having a global knowledge about all agents is likely required, which on the other hand restricts the ability of building a self-organized and scalable system.

Differently from the works mentioned above, the approach presented in this paper is a self-organized, scalable and adaptive approach for robot swarms in which robots work in a full autonomy. Neither building instructions nor global knowledge are available and the quality of the constructed wall is assessed using the above-mentioned set of performance (quality) metrics.

IV. The Construction Behavior

In this section, we describe the behavior of the individual robots that emerges in constructing walls to isolate areas for preventing undesired access. The design starts from the microscopic level (individual level) at which we define the rules applied by the robots and ends up at the macroscopic level (swarm level) at which we measure the overall performance. Characterizing the link between the microscopic and the macroscopic behaviors is one of the main well-known challenges in swarm robotics, in general, and particularly in this task, where the macroscopic behavior is restricted to specific criteria. In the following, we are presenting two behaviors: the general construction behavior and the adaptive construction behavior.

A. The general construction behavior

In the general construction behavior, we focus on designing a robot behavior that leads to construct a wall for isolation in an unknown environment. Robots which are applying the general construction behavior perform the following tasks: searching for building blocks; gripping the found blocks; navigating to the construction area; and finally unload the blocks and tune the wall before they start again to search

 ISBN: 978-1-61208-463-3

for new building blocks. Figure 2 shows the a finite state machine (FSM) that describes the behavior of the individual robot on the microscopic level. The robot starts initially in the searching state. In this state, it wanders randomly around the part of the arena at which the building blocks are scattered trying to find a block. Robots are required to avoid gripping blocks that are already gripped by other robots (e.g. using broadcasts). As soon as a block is found, the robot comes closer to the block attempts to grip it. In case the gripping was successful, the robot starts moving towards the construction area to place the building block. While moving, robots avoid other robots and objects using both their camera and proximity sensors. As soon as the robot reaches the construction area, it aims to unload the building block at the best possible position, taking into consideration other building blocks that are already unloaded. Achieving an efficient unload that meets the quality measures defined above is a non-trivial task. In the implemented construction behavior, we allow the robots to exploit the moveable gripper for adjusting the unloading position so that the distance to the nearest block with no two direct neighbors is minimized. This behavior allows to increase the compactness of the constructed wall and decrease its thinness. After finish unloading the building block, the robot switches to a special state referred to as the wander state. This state is identical to the searching state, except that robots at this state are not allowed to grip building blocks. This state is activated each time a robot unloads a building block and while the robot is moving near to the construction area for preventing the robots to grip blocks that are already placed as a part of the wall. The robot stays in the wander state for a particular time before it switches back to the standard searching state and becomes again able to grip blocks.

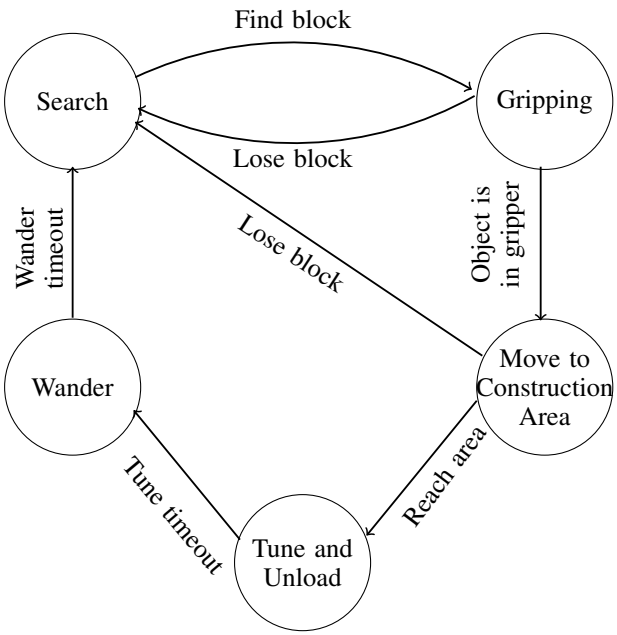


Figure 2. The finite state machine of the robots' construction behavior.

B. The adaptive construction behavior

General construction behavior allows for the construction of a structure (here a wall) at a desired location of the arena. However, it does not include the recognition of the construction dynamics. Therefore, we extend the general construction behavior to the adaptive construction behavior, in which robots tune their behaviors in respect to the dynamics of the construction task. The absence of building instructions and the dynamics of the construction process belong to the main challenges of a construction task. The main difficulty for the individual robots would be to recognize the degree of progress achieved at the macroscopic level. However, recognize this macroscopic feature and adopt the individual behavior accordingly to it allows the swarm to cope with the dynamics of the task and thus perform it more efficiently. Figure 3 shows the bi-directional link between the microscopic behavior of the individuals and the macroscopic performance of the swarm.

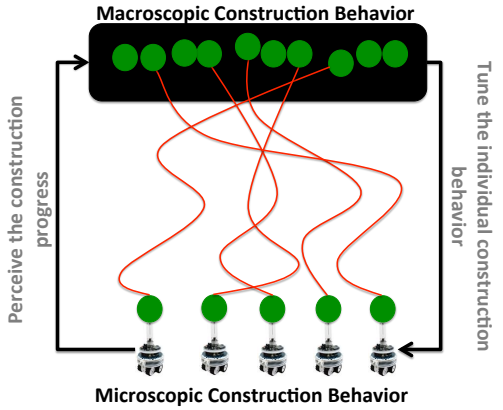


Figure 3. The micro-macro link in the construction task.

In order to recognize the dynamics of the construction task at the microscopic (individual) level, robots measure the time required to unload their building blocks at the construction area. This time is referred to as the *unload-time* – it starts from the moment the robot arrives carrying the building block to the construction area up to the moment the robot unloads the block successfully. It includes the time spent by the robot looking for a suitable location for unloading the block. While the construction of the wall proceeds, it becomes more difficult for the robots to find a free location to unload their building blocks and hence longer unload-time is experienced. Longer the time the robot spends in searching for a suitable location to unload, higher the probability is of having the progress of the construction at an advanced stage. Consequently, less blocks need to be transported to the construction area and less robots are required for continuing the task. Based on that, in the adaptive construction behavior robots can be in one of the following states: the *constructing* state or the *resting* state. At the constructing state the robot participates in the building activities captured in Figure 2. Whereas, the resting state is selected when the robot decides to leave the construction task for the moment. The switch between these two states is performed probabilistically. The probability to switch from the constructing state to the resting state is denoted by $Prob_r$ and the probability of switching from the resting state to the constructing state is denoted by $Prob_c$.

$$Prob_c = 1 - Prob_r$$

During the task execution, a robot which is at the constructing

state switches to the resting state with the probability $Prob_r$ or stays in the constructing state with the probability $Prob_c$. Similarly, a robot at the resting state switches to the constructing state with the probability $Prob_c$ or stays at the resting state with the probability $Prob_r$. Figure 4 illustrates the two states and the different switching probabilities.

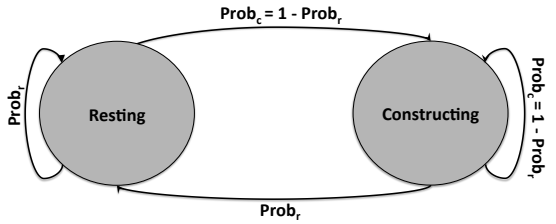


Figure 4. Robot's states under the adaptive construction behavior.

Let us observe the particular transport of the i-th building block by a robot. We use Δt_i to denote the time required to unload the i-th block. This is the unload-time of the i-th block and it is measured independently by the robot itself. The unload-time of the previous block is denoted by Δt_{i-1}. If the robot has the following condition true:

$$\Delta t_i >= \Delta t_{i-1} + \alpha$$

where α is a design parameter, it means that the latest unload-time was longer than the previous unload-time with the period α. Thus, the robot starts to contact its local neighbors in order to inform them that the unload action is getting more difficult. This indicates a specific progress in the construction process. Therefore, less blocks maybe required and consequently less robots need to participate on the building task. Each robot that receives the message increases its resting probability $Prob_r$ as follows:

$$Prob_r = min\{Prob_r + \Theta_r, 1\}$$

where Θ_r is the probability increment step and it represents a design parameter.
However, when the robot perceives a shorter unload-time than the previous one meeting the following condition:

$$\Delta t_i < \Delta t_{i-1} + \alpha$$

It is interpreted at the individual level as having an easier unload action than the previous one. This can be the result of having the construction task at an early stage, or having a gap in the constructed wall that needs to be filled with blocks. It can be also an indication of a new section in the arena that needs to be isolated by constructing a new wall. Whatever the reason is, the robot starts to broadcast this information to its local neighbors announcing the increasing need to participate on the construction activities. This message decreases the resting probability ($Prob_r$) and increases the constructing probability accordingly, as in the following:

$$Prob_r = max\{Prob_r - \Theta_c, 0\}$$

where Θ_c is the probability increment step and it represents a design parameter. In this paper, we assume to have $\Theta_c = \Theta_r$ The adaptive construction behavior can result in the following improvements:

- Recognizing the dynamics of the macroscopic construction performance and acting accordingly.
- Providing a resource-efficient solution: by using the necessary amount of building blocks and preserve the rest for other construction tasks.
- Providing a robot-efficient solution: by using the number of robots required currently. This leads to preserve robots for joining new construction tasks.

Robots that apply the adaptive construction behavior exploit both indirect communication (stigmergy) and local communications with their direct neighbors. Stigmergy can be observed in the interpretation of the length of the unload-time by the individual robots. This time is directly related to actions taken by other robots, namely, to previous unload actions. Hence, it represents a piece of information transferred using the physical environment. Direct communication is used, on the other hand, to inform the local neighbors about the difficulty of the unload action in order to enable them to take an appropriate decision concerning their next state.

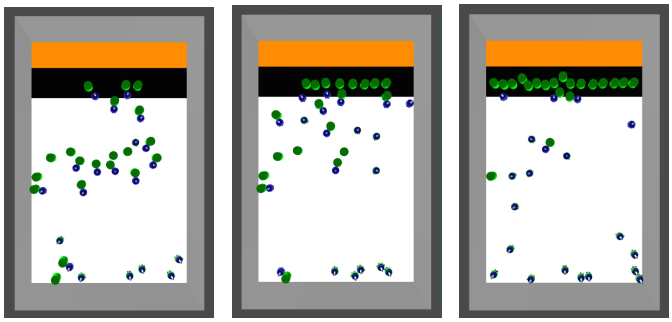
V. Experiments and Discussion

In this section, we present a set of physics-based simulations in order to verify our approaches performed using the state-of-the-art simulator ARGoS [15]. ARGoS is an efficient simulator that allows to simulate large swarms of robots with taking the desired level of physical details into consideration. In our swarm, we are using foot-bots – wheeled robots equipped with proximity sensors, cameras, a gripper, and a range and bearing system. We consider a $6 \times 4\ m^2$ working arena that is divided in three parts: a $0.8 \times 4\ m^2$ undesired part (a dangerous section) which is depict in orange at the top of the arena, a $4.5 \times 4\ m^2$ safe section that is depicted in white, and the $0.7 \times 4\ m^2$ construction area. The construction area is indicated by a black strip on the floor in addition to a set of lights that helps navigating the robots at the working arena. The robot navigation is performed as a combination between random walks and attraction and repulsion to the lights deployed in the environment. In real-world scenarios, the construction area could be indicated with other environmental parameters such as light density, humidity, or others. We are using cylinder building blocks, each with a diameter of $0.2\ m$, thus 20 building blocks are enough to construct the desired wall. Initially, both robots and building blocks are scattered uniformly at the safe section of the arena and the positions are chosen at random for each new experiment. We set the running time of each experiment to 3000 seconds. Furthermore, two configurations of the construction experiments are used, one with 20 building blocks and the other is with 60 building blocks.

As mentioned above, 20 building blocks is the amount sufficient to build the desired wall based on the dimensions of the constructions area. We first allow the robots to use the general construction behavior and afterwards the adaptive construction behavior. A set of snapshots at different time steps during a particular trail of the construction process are depicted in Figure 5 and in Figure 6. As we can notice, both behaviors perform well in constructing the wall according to the required criteria. However, the remarkable difference in the performance between the general and the adaptive behavior can be seen in the experiments in which 60 building blocks are used. Figure 7 shows the progress of the construction when robots are using the general construction behavior. As we can see, robots in this case are not able to recognize the progress of the construction

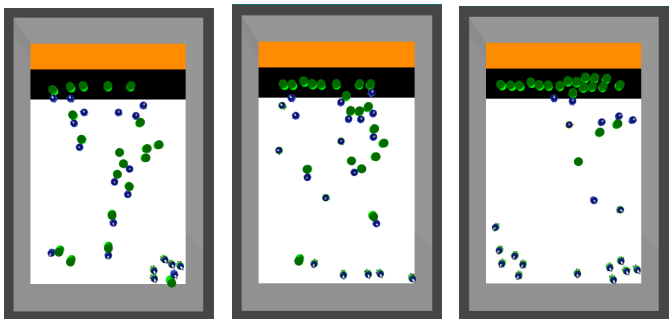
 ISBN: 978-1-61208-463-3

process over time. Thus, they are not reacting in an adaptive manner to the dynamics of the construction task. Therefore, they continue to use building blocks as long as they are available or up to the end of the experiment (i.e., second 3000). This leads to the thick structure we can observe in Figure 7c. On the contrary, when robots use the adaptive construction behavior, they become able to recognize and adapt to the dynamics of the construction task. This is what we can observe clearly in Figure 8, in which robots stop to participate on the construction task over time affected by the feedback given by other robots about the increasing difficulty of unloading a building block. The improvement in the quality of the obtained wall is clear in Figure 8c. Moreover, Figure 8 depicts no increment in the mass of the wall after the time step 2000. This means that robots applying the adaptive construction behavior become much earlier available to join other tasks than robots applying the general construction behavior, thanks to the ability of recognizing the end of a construction task.



(a) At t.s. 300 (b) At t.s. 500 (c) At t.s. 1000

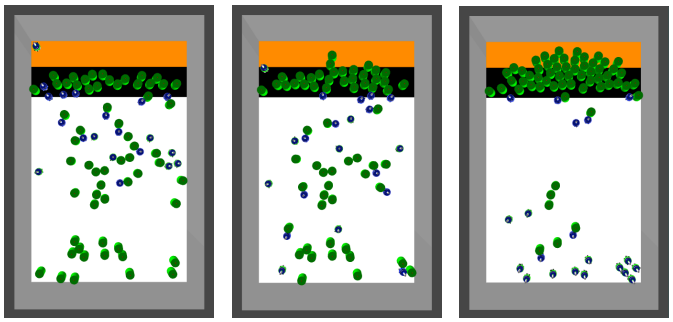
Figure 5. Snapshots of the wall construction task using 20 blocks at different time steps. Robots are applying the general construction behavior.



(a) At t.s. 300 (b) At t.s. 500 (c) At t.s. 1000

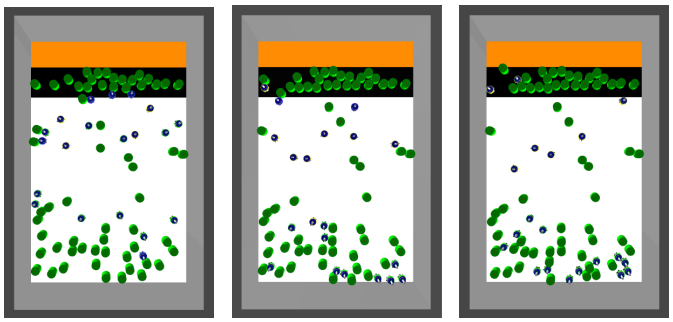
Figure 6. Snapshots of the wall construction task using 20 blocks at different time steps. Robots are applying the adaptive construction behavior.

We have repeated each of the 4 experiments, presented above, 30 times and the quality measures of the obtained walls in addition of the number of used blocks were averaged. Figure 9 shows the number of building blocks used in constructing the desired wall (We depict both the mean and the standard deviation). In Figure 9a, the available amount of blocks is 20 and we can observe that the amounts used by both the general and the adaptive behavior are similar. Whereas, the amount



(a) At t.s. 1000 (b) At t.s. 2000 (c) At t.s. 3000

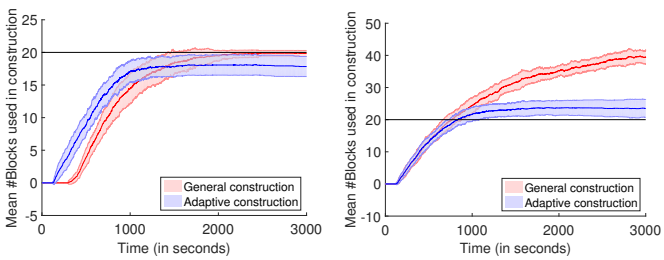
Figure 7. Snapshots of the wall construction task using 60 blocks at different time steps. Robots are applying the general construction behavior.



(a) At t.s. 1000 (b) At t.s. 2000 (c) At t.s. 3000

Figure 8. Snapshots of the wall construction task using 60 blocks at different time steps. Robots are applying the adaptive construction behavior.

used by the general behavior diverges significantly from the one used by the adaptive behavior when 60 blocks are used. This what we can see in Figure 9b, in which the amount of blocks used by the adaptive behavior stays near the sufficient amount.

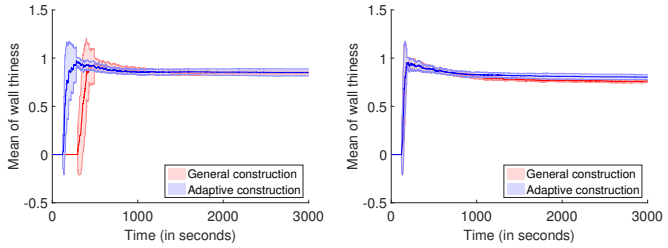


(a) 20 building blocks. (b) 60 building blocks.

Figure 9. The number of building blocks used by both the general and the adaptive construction behaviors. (a) when 20 blocks are used, (b) when 60 blocks are used.

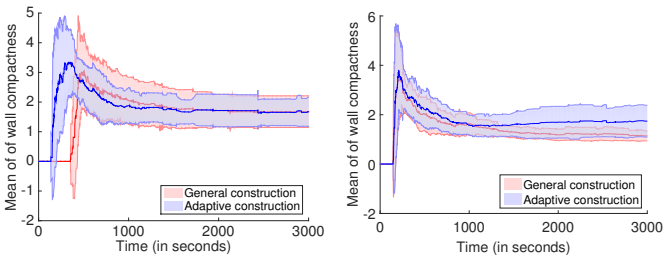
The quality metrics of the wall were averaged over 30 runs of the different experiments and as we can see in Figure 10, Figure 11, and Figure 12 both behaviors: the general and the adaptive have achieved walls with a high quality and with near-to-optimal thinness, compactness and coverage.

 ISBN: 978-1-61208-463-3



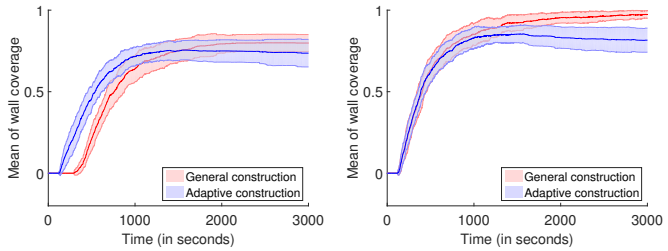
(a) 20 building blocks. (b) 60 building blocks.

Figure 10. The wall thinness achieved by both the general and the adaptive construction behaviors. (a) when 20 blocks are used, (b) when 60 blocks are used.



(a) 20 building blocks. (b) 60 building blocks.

Figure 11. The wall compactness achieved by both the general and the adaptive construction behaviors. (a) when 20 blocks are used, (b) when 60 blocks are used.



(a) 20 building blocks. (b) 60 building blocks.

Figure 12. The wall coverage achieved by both the general and the adaptive construction behaviors. (a) when 20 blocks are used, (b) when 60 blocks are used.

VI. Conclusion

The self-organized construction behavior is a main part of a wide range of tasks at which swarm robotics are planned to be deployed in the future. Therefore, it represents an attractive research challenge, specifically when no building instructions are known a priori or coded in the system. In this paper, we first define a set of quality criteria of the desired wall. Afterwards, we propose two construction behaviors that can be applied by a swarm of simple and homogenous robots. The first is the general construction behavior, which includes a set of simple rules that allow the robots to wander in an unknown arena, search for building blocks, grip them and navigate to where they should be placed. One of the main challenges, here, is to unload the building block properly such that the emergent wall respects the given quality metrics. The general behavior performs optimal, when only the sufficient amount of building blocks is available or when the time of the construction task is set to be the required one. The adaptive construction behavior, on the other hand, is designed using a probabilistic approach to cope with the dynamics of the construction task. It allows the robots to recognize the progress of the construction task over time and to act accordingly. The adaptive construction behavior achieves high quality walls similar to the general construction behavior. However, it both avoids the use of unnecessary building blocks and sets the robots earlier free for participating on other tasks through realizing the end of the current construction task.

References

[1] N. Franks, A. Wilby, B. Silverman, and C. Tofts, “Self-organizing nest construction in ants: sophisticated building by blind bulldozing,” Animal behaviour, vol. 44, 1992, pp. 357–375.

[2] E. Bonabeau, “Editor’s introduction: stigmergy,” Artificial Life, vol. 5, no. 2, 1999, pp. 95–96.

[3] G. Beni, “From swarm intelligence to swarm robotics,” in Swarm Robotics. Springer, 2005, pp. 1–9.

[4] G. Theraulaz, J. Gautrais, S. Camazine, and J. Deneubourg, “The formation of spatial patterns in social insects: from simple behaviours to complex structures,” Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, vol. 361, no. 1807, 2003, pp. 1263–1282.

[5] E. Bonabeau, G. Theraulaz, J. Deneubourg, S. Aron, and S. Camazine, “Self-organization in social insects,” Trends in Ecology & Evolution, vol. 12, no. 5, 1997, pp. 188–193.

[6] O. Bruinsma and L. Wageningen, An analysis of building behaviour of the termite Macrotermes subhyalinus (Rambur). Landbouwhogeschool te Wageningen Wageningen, 1979.

[7] E. Bonabeau, G. Theraulaz, J. Deneubourg, N. Franks, O. Rafelsberger, J. Joly, and S. Blanco, “A model for the emergence of pillars, walls and royal chambers in termite nests,” Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 353, no. 1375, 1998, pp. 1561–1576.

[8] M. Belić, V. Škarka, J. Deneubourg, and M. Lax, “Mathematical model of honeycomb construction,” Journal of mathematical Biology, vol. 24, no. 4, 1986, pp. 437–449.

[9] R. Brooks, P. Maes, M. Mataric, and G. More, “Lunar base construction robots,” in IEEE International Workshop on Intelligent Robots and Systems ’90. ’Towards a New Frontier of Applications’, Proceedings. (IROS). IEEE, 1990, pp. 389–392.

[10] C. Melhuish, J. Welsby, and C. Edwards, “Using templates for defensive wall building with autonomous mobile ant-like robots,” in In Proceedings of Towards Intelligent and Autonomous Mobile Robots, 1999.

[11] J. Wawerla, G. Sukhatme, and M. Mataric, “Collective construction with multiple robots,” in IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3. IEEE, 2002, pp. 2696–2701.

[12] C. Parker, H. Zhang, and C. Kube, “Blind bulldozing: multiple robot nest construction,” in IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), vol. 2. IEEE, 2003, pp. 2010–2015.

[13] E. Bahceci, O. Soysal, and E. Şahin, “A review: Pattern formation and adaptation in multi-robot systems,” Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-43, 2003.

[14] J. Fredslund and M. Mataric, “A general algorithm for robot formations using local sensing and minimal communication,” IEEE Transactions on Robotics and Automation, vol. 18, no. 5, 2002, pp. 837–846.

[15] C. Pinciroli, V. Trianni, R. OǴrady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Di Caro, F. Ducatelle, M. Birattari, L. Gambardella, and M. Dorigo, “ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems,” Swarm Intelligence, vol. 6, no. 4, 2012, pp. 271–295.

 ISBN: 978-1-61208-463-3

Adaptive Embedded Control for a Ball and Plate System

Jun Xiao

Informatics Institute
University of Amsterdam
The Netherlands
Email: J.Xiao@uva.nl

Giorgio Buttazzo

Real-Time Systems Laboratory
Scuola Superiore Sant'Anna
Pisa, Italy
Email: giorgio@sssup.it

***Abstract*—In the application of embedded control systems, timing is an important factor for ensuring a desired system performance. However, the time properties are difficult to manage under the constraints of a given hardware platform and the application tasks have large computation time variations. These require the usage of adaptive mechanisms on several aspects of the control system, from sensory inputs to actuation outputs. Taking a control system consisting of ball and plate device as an example, this paper shows how to manage time and achieve performance improvements by using adaptive strategies in sensing and control. Sensory acquisition, control, and actuation are performed on a STM32F4 microcontroller. A camera mounted on the embedded board is used to detect the ball position. To cope with the limitations of computational speed of the embedded board, an adaptive approach that changes the capture area of the image acquisition process at every execution is proposed for improving the detection accuracy. A system simulator using a TrueTime kernel that provides multitasking environment has been developed as a support tool for designing the controller. Jitter compensation techniques, which adaptively update the control parameters at each sampling instant, are adopted to deal with the performance degradation due to the sampling jitter. Conditions for finding the set of proper control parameters, a practical issue of applying jitter compensation techniques, are also presented. Finally, the system has been implemented and tested on top of the Erika Enterprise real-time kernel.**

Keywords–Adaptive Embedded Control Systems; Camera Detection; Jitter Compensation; PID Control.

I. INTRODUCTION

Balancing is one of most challenging issues in the control field. There are lots of platforms for studying control algorithms for system balancing such as the ball-beam system and the inverted pendulum. Among those, the ball-and-plate system consists in controlling the angular position of a plate with two degrees of freedom (pitch/roll) in order to keep a ball always in the center of the plate in the presence of disturbances. When the ball starts moving, it will roll off the end of the plate if no control action is taken.

The first major challenge is to sense the ball position accurately and in a non-cumbersome, yet inexpensive way. In some implementations, a touch screen is used as the ball position sensor [1], whereas in most cases a Charge-coupled Device (CCD) camera is quite popular for ball detection. Christoph H. Lampert, et al. [2] proposed to overcome the computational bottleneck by making use of the massively parallel architecture of a Graphics Processing Unit (GPU) in a modern computer graphic card. This solution, however, is quite expensive. The second consideration is about the effectiveness of different control algorithms. Nonlinear control methods like Back-stepping control were successfully applied by Lin, et al, in [3] and Ker, et al, in [4]. In recent works [5] [6], intelligent control skills, such as fuzzy logic control, are also studied. However, most of these control methods work only if the sampling period of the control system is small, which strongly requires high speed processing hardware including sensor and microcontroller.

In this paper, sensory acquisition and control of the ball-and-plate system are performed by the STM32F4 microcontroller. A camera mounted on the embedded board behaves as sensor to detect the ball position. Under such hardware constraints, we propose an adaptive detection procedure to cope with the limited capacity of Random-Access Memory (RAM) storage and limited computational speed of the given embedded board. The proposed control strategy includes a jitter compensation technique and is adopted to deal with the performance degradation due to the sampling jitter introduced by the multitasking environment in the embedded board.

The rest of the paper is organised as follows. Section II presents an overview of the system architecture. Section III describes the decision of the sampling period and the adaptive camera detection procedure. Section IV analyzes the Proportional-Integral-Derivative (PID) control algorithm, the effect of the sampling jitter and the jitter compensation techniques. Simulation results and system testing are shown in Section V. Section VI draws the conclusions.

II. SYSTEM DESCRIPTION

The ball-and-plate control system considered in this work consists of two embedded boards: one board (referred to as board A hereafter), equipped with a camera, detects the ball, computes its position, and sends this information to the other board (board B) through a wireless network. Board B receives the data, applies the control algorithm and supplies a proper voltage to the servo motor for actuating the plate. The illustrative overview of the system is shown in Figure 1.

In addition, a system monitor has been developed under the Linux operation system to provide a graphic user interface for displaying the system state. The monitor records the last ten ball positions and plots dynamically the control error with respect to the reference. The monitor also allows users to modify and to update the controller parameters of the system at run time. The monitor communicates with board B through

 ISBN: 978-1-61208-463-3

a serial port. The design and implementation of the Computer-Monitor are out of the scope of this paper.

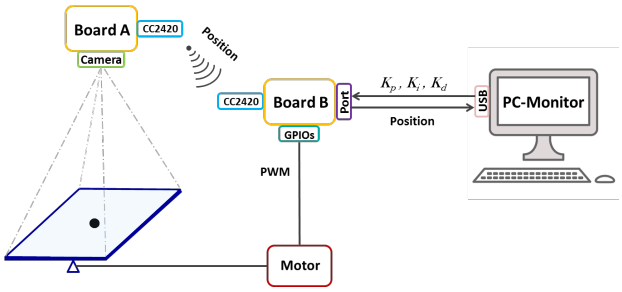


Figure 1. Overview of the ball-and-plate system.

The hardware boards used to control the ball-and-plate system are based on the STM32F4 platform and its expansion boards, CC2420 radio transceivers.They are briefly described in the following paragraph.

STM32F4 discovery board is a microcontroller featuring 32-bit ARM Cortex-M4F core, 1 MB Flash, 192KB RAM. The STM32F4 discovery board has up to 14 timers that can generate the Pulse Width Modulation (PWM) signals to control the servo motors [7].

Three boards extend the Discovery capabilities. The first board is a base board. It provides connectors with General Purpose Input Outputs (GPIOs), connector for camera board and connector for Liquid Crystal Display (LCD) board. The second board is a LCD board. It has a 3"5 display to visualize information such as the ball position and the controller parameters. The last board is a camera board. The resolution of the camera is set to be QQVGA (160×120). It is not possible to obtain an image with a higher resolution otherwise the image to be stored would exceed the RAM size (192KB).

Communication between board A and board B is realized by the CC2420, a true single-chip 2.4 GHz IEEE 802.15.4 compliant transceiver designed for low-power and low-voltage wireless applications. The CC2420 is connected with the base board. The wireless protocol adopts the IEEE 802.15.4 standard. The wireless personal area networks (LR-WPANs) is implemented by using the μ Wireless stack.

III. Detection Procedure

Ball detection is done based on the frame taken by the camera at the beginning of each sampling period. Accurate detection results can be obtained by processing all image pixels, but this comes at a cost of a high computational effort, especially when using a microcontroller with a limited computational speed. The execution time due to image processing determines the minimum sampling period of the control system, hence a compromise has to be found between the limit imposed by the computational speed of the STM32F4 board and the controller requirement of running at a faster sampling rate.

A. Image color data

The frame taken by the camera is expressed internally by a matrix $M_{m \times n}$ containing pixels. The size of the matrix $m \times n$ is determined by the image resolution. The pixels are encoded using a Red-Green-Blue (RGB) 16-bit model. For each element m_{ij} in the M, 5, 6 and 5 bits are used to represent the intensities of red, $green$ and $blue$ component, respectively. Thus, the values m_{red} and m_{blue} representing red and blue intensities are integer numbers ranging from 0 to 2^5, while for green m_{green}, the range is 0-2^6.

B. Computation of threshold

To facilitate the detection procedure, we used a dark ball on a white plate. Hence, the ball is detected by considering the pixels whose color levels are below certain thresholds. Each color component, namely red, $green$, $blue$, has one threshold, donated as THS_{red}, THS_{green} and THS_{blue} respectively. If the values of red, $green$ and $blue$ extracted from one pixel are all smaller than their corresponding thresholds, that pixel is considered to belong to the ball.

Although image thresholding greatly simplifies the detection procedure, the result heavily depends on the light condition of the environment. The darker the environment, the lower the thresholds, and vice versa. In order to make the detection procedure more adaptive, the threshold is automatically computed by the system during an initial calibration procedure. In more details, board A takes a frame of the whole plate with the ball before the start of the system. Then the red intensity m_{red} is extracted from every pixel. For each possible value v (v=0,1,2,...,31), it counts the total number of pixels N_v^{red} whose m_{red} equals v. Finally, the threshold THS_{red} is determined by the following formula:

$$THS_{red} = 0.9 \times Index_{0-14}^{red} + 0.1 \times Index_{14-31}^{red}, \quad (1)$$

where:
$Index_{0-14}^{red} = k$ such that $N_k^{red} = \max N_i^{red}, i = 0, 1, ..., 14$,
$Index_{15-31}^{red} = k$ such that $N_k^{red} = \max N_i^{red}, i = 15, 16, ..., 31$.
The same steps is applied for the calculation of green and blue thresholds THS_{green}, THS_{blue}.

C. Sampling period

By comparing each element in the frame matrix M with the thresholds, we can transfer M to a matrix L containing logical values l_{ij}, where a value of 1 means the pixel corresponds to the ball.

Once the pixels belonging to the ball are identified, a $center\ of\ mass$ [14] algorithm is used to estimate the ball position. Based on the resulting matrix L, it records the coordinates of each black pixel (i, j) and computes the total number of dark pixels N. Then, the ball position is determined by computing the average coordinate of all black pixels. Specifically, a region to be scanned is delimited by the upper left pixel coordinating (x_{min}, y_{min}) and the lower right pixel coordinating (x_{max}, y_{max}). The ball position (X, Y) is computed by the following formulas:

$$N = \sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} l_{ij}, \quad (2)$$

$$X = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} i \times l_{ij}}{N}, \quad (3)$$

$$Y = \frac{\sum_{i=x_{min}}^{x_{max}} \sum_{j=y_{min}}^{y_{max}} j \times l_{ij}}{N}. \quad (4)$$

 ISBN: 978-1-61208-463-3

A test program was written to estimate the execution time of finding the ball position by scanning every pixel. It takes around 2.4 seconds to finish the computation on the STM32F4 microcontroller. As a sampling period, 2.4 seconds is too long for controlling the ball-and-plate system.

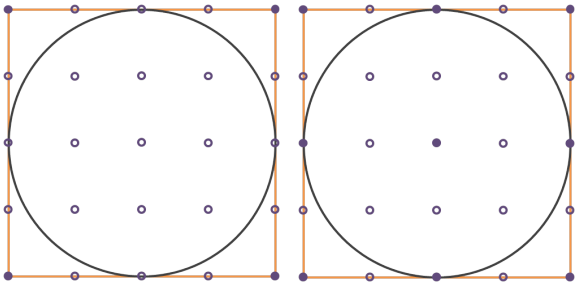
One way to reduce the sampling period is to scan less pixels. Considering the fact that the ball is in a square covered by several pixels, (in general $p \times q$ pixels and 4×4 in our case), one can scan only one pixel among every n_x and n_y ($n_x \leq p, n_y \leq q$) consecutive pixels along the two directions. Thus the total number of points N_{scan} to be scanned is reduced to $\frac{m \times n}{n_x \times n_y}$.

If only one pixel among four consecutive pixels is selected to be scanned, the execution time of detection decreases to around 0.15 seconds. Considering also the time needed for the execution of other tasks in board A, the sampling period of the control system was set at 0.2 seconds to avoid overload during the execution.

D. Adaptive scan

As mentioned above, to reduce the computation time of the scan, the basic detection procedure reads one pixel every four consecutive pixels. This is called constant capture area procedure because the capture area keeps always the whole plate, in other words, x_{min}, y_{min}, x_{max}, y_{max} are constant at each sampling period.

This procedure, however, fails to detect the ball if the scanning points are exactly the vertexes of the square covering the ball, as shown in Figure 2(a), where the full dots stand for the pixels scanned. Therefore, it is risky to adopt this method as it may lose detection in some occasions.



(a) (b)

Figure 2. Ball coverage and scanning points.

To improve the detection rate, pixels need to be scanned more intensely without increasing N_{scan}. Noticing that most of computation is wasted for scanning the pixels unrelated to the ball, the idea of $region\ of\ interest$ [14] is used to reduce the computational cost by searching for the ball only in a smaller region of interest, moving with the ball.

Thus, we shrink the capture area for catching the ball at each sampling period. The values representing the upper left (x_{min}, y_{min}) and lower right (x_{max}, y_{max}) corner are updated at each sampling period due to ball movement. Specifically, the next focusing area is a square centered by the last ball position and composed of $\frac{m \times n}{s_x \times s_y}$ pixels, where s_x and s_y (2×2 in our case) are the reduction factors. The matrix size of the focusing area is $\frac{1}{s_x \times s_y}$ of matrix M. Therefore, one pixel among every $\frac{n_x \times n_y}{s_x \times s_y}$ pixels can be scanned without increasing the time for the frame processing, as shown in Figure 2(b).

However, the second solution is based on the assumption that the ball moves slowly such that at the next sampling time, it is always inside the scope of the region of interest. Otherwise, the ball will not be detected.

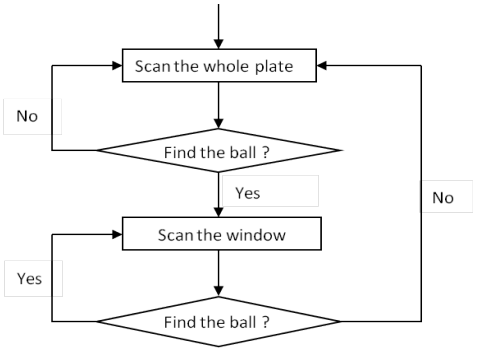


Figure 3. Block diagram for the adaptive detection procedure.

To improve detection, the two solutions are combined to overcome their drawbacks. The adaptive detection procedure is called by adaptive capture window, as shown in Figure 3. The detection starts from executing the procedure in a constant capture area. Once the ball is detected, the detection switches to scan the adaptive window, which is the square centered by the last detected ball position. If the ball moves out of the region of interest, the detection goes back to the procedure of constant capture area.

IV. Control Design

This section describes how the controller has been designed to meet the performance specifications and how the jitter compensation techniques can be applied for improving the control performance to deal with the performance degradation due to the sampling jitter. Considering that the mutual interference between the two actuation axes, it is possible to design two independent controllers for the two (x and y) axes. The following analysis focuses on one direction, but can be applied to both axes.

A. Control performance metrics

The primary criterion for evaluating the performance of control systems is to meet stability requirements and response time specifications. But these criterions do not provide a universal value to judge the control performance. We use the notion of $Quality\ of\ Control$ (QoC), called performance rate function, to evaluate the control performance of closed-loop systems. The QoC in [11] is defined as:

$$QoC(T_0, t) = \frac{1}{IAE(T_0, t)}, \tag{5}$$

where:

- IAE is the integral of the absolute system error, which is the difference between the desired response of the system $y_{ref}(t)$ and its actual response $y_{act}(t)$.

$$IAE = \int_0^\infty |y_{act}(t) - y_{ref}(t)| dt. \tag{6}$$

- $IAE(T_0, t)$ denotes the IAE value obtained by a controller designed with a nominal period T_0, but running with a period t.

B. System model

The full derivation of the system state model is out of scope of this paper; hence, this part only shows the resulting system model.

The linearized relationship between the rotation angle of the motor α and the rotation angle of the plate θ is:

$$\theta = \frac{-68.28(\alpha - 0.49)}{194.28(\alpha - 0,49) - 141.52}. \quad (7)$$

The transfer function of the servo motor having the desired rotation angle α_d as input and the actual rotation angle α_{act} as output is:

$$G(s) = \frac{\alpha_{act}(s)}{\alpha_d(s)} = \frac{1}{0.002s + 1}. \quad (8)$$

The state space form of dynamic model of the ball movement is:

$$\dot{x}_1 = x_2, \quad (9)$$

$$\dot{x}_2 = \frac{3}{5} g \sin\theta, \quad (10)$$

where x_1 represents the ball position of x-axis, x_2 denotes the speed of the ball.

C. Control design

1) PID control law: The PID control [12] algorithm using the closed-loop feedback mechanism is commonly applied in the industrial control systems.

Since the sampling period T is fixed to be $0.2s$, we directly design the PID controller in the discrete time domain. The input to the controller is the ball position error with respect to the reference $e(k)$ at time kT, the output of the controller is the rotation angle of motor $\alpha(k)$ at time kT.

Thus, the discretized form of the PID controller is:

$$\alpha(k) = K_p e(k) + K_i T \sum_{i=0}^{k} e(i) + K_d \frac{e(k) - e(k-1)}{T}. \quad (11)$$

A system simulator is developed to help in controller design. By tuning the three control parameters in the simulator, when $K_p = 10$, $K_i = 0.5$, $K_d = 15$, the system is stable, meeting the control performance.

2) Multitasking and jitter compensation: Nowadays, multitasking is common in real applications. More specifically, in embedded control applications, usually there are multiple tasks executed concurrently in one processor. In particular, in the ball-and-plate implementation, besides the control task, there are other periodic tasks to toggle LEDs to indicate system running state, to print data such as PID parameters on the LCD screen.

At each execution of the discretized PID control task, the output is computed based on the value of three parameters (K_p, K_i, K_d). The three parameters are constant during the execution of each job in a traditional PID control task, assuming that the sampling is performed at equidistant sampling time instants.

However, the assumption is not realistic due to the existence of sampling jitter. In fact, we suppose that there are two additional tasks τ_1 and τ_2 with execution times $5ms$ and $15ms$ and periods $100ms$ and $150ms$ running concurrently with the control task τ_3 including both the work of sensory detection and control algorithm execution in the microcontroller. If the task set composed of τ_1, τ_2, τ_3 is scheduled by Rate Monotonic, the task scheduling of one hyperperiod is illustrated in Figure 4. From Figure 4, it is easy to see that

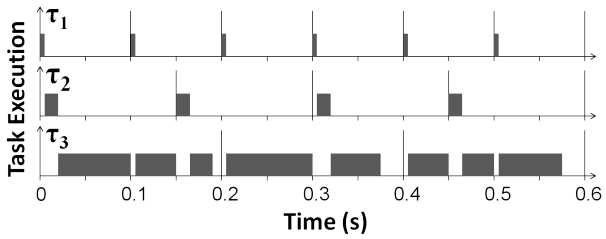


Figure 4. Task scheduling of one hyperperiod by rate monotonic.

the start times $s_{i,k}$ of the k^{th} job in the control task i are not always $nT(n = 0, 1, 2...)$. The difference between the start times (relative to the request times) of two or more instances of a periodic task introduces sampling jitter. The set of all possible sampling jitters, $SJ(\tau_i)$, for task τ_i is defined as [13]:

$$SJ(\tau_i) = \{h_{i,k} | h_{i,k} = s_{i,k+1} - s_{i,k}, k = 0, 1, 2, ...\}. \quad (12)$$

Sampling jitters that will appear at the run time of control task τ_3 can be analyzed offline. The jitters of the first three jobs repeat since the schedule repeats every hyperperiod. Therefore, $SJ(\tau_3) = 0.185, 0.2, 0.215$.

As integral (K_i) and the derivative (K_d) actions depend on the time interval between the last and current sampling time, the actual control performance of the controller running within the multitasking environment is jeopardized by the sampling jitter. The jitter compensation technique updating the PID controller parameters based on the sampling jitter, proposed in [15], can be applied to improve the control performance.

3) Setting the control parameters: To apply jitter compensation, the PID parameters for all possible jitters have to be chosen at the design phase. However, as a practical issue, finding a proper set of parameters is not easy since a bad composition of PID parameters would make the system response even worse.

Two conditions to judge whether the set of PID parameters is proper is based on the performance-rate functions. Specifically, for a control task τ, $\forall T_0 \in SJ(\tau)$, $\forall T_0' \in SJ(\tau)$, and $T_0 \neq T_0'$, if the QoC obtained by running the PID controllers prepared for the compensation satisfies the following two conditions:

- if $T_0 < T_0'$, $QoC(T_0, T_0) > QoC(T_0', T_0')$,
- $QoC(T_0, T_0') < QoC(T_0, T_0)$ and $QoC(T_0', T_0) < QoC(T_0, T_0)$,

then the PID parameters set is proper.

The first condition derives from the fact that smaller sampling periods lead to better QoC. The second item requires the optimal selection of the PID control parameters such that for a specific sampling period, the best QoC is produced only when the actual execution period of the control task equals the designed sampling period.

 ISBN: 978-1-61208-463-3

V. EXPERIMENTAL RESULTS

In this section, we illustrate both simulation results and system testing.

A. Simulation results

1) Simulation model: To carry out the simulation experiments, a system model has been developed in Simulink [10]. Figure 5 is a general view of the whole model. In this model, the block *motor* functioning is defined by (8), which describes the transfer function of a servo motor. The *plate* block represents the plate rotation performed by (7) .The dynamic model (9) and (10) is simulated by the *rolling ball* block.The tasks execution (three periodic tasks including control task τ_3, together with τ_1 and τ_2, discussed in Section IV-C2.) is simulated using the $TureTime\ Kernel$ block, provided by TrueTime [8], which facilitates co-simulation of controller task execution in real-time kernels. The two detection procedures discussed in Section III-D are also implemented.

2) Camera detection: Camera detection results and corresponding system responses results by using the adaptive capture window and the constant capture area are compared in Figure 6 and Figure 7, respectively. In Figure 6, 1 stands for the successful detection, while 0 represents missing detection at the current execution.

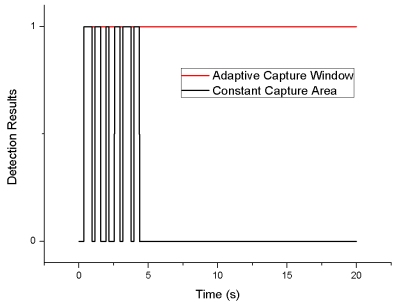


Figure 6. Comparison of the camera detection results using two different detection procedures.

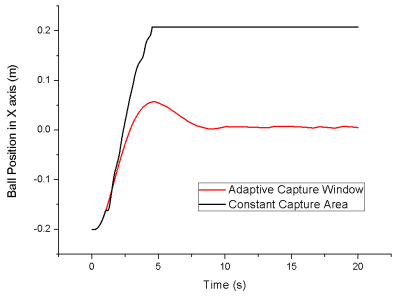


Figure 7. Comparison of system responses using two different detection procedures.

Experiments showed that the system was unstable using the constant sampling area, since the ball was no more detected after few seconds and stayed near the beam. On the other hand, the experiments showed that the system became stable using the adaptive capture window, due to the higher achievable detection rate.

3) Jitter compensation: Following the conditions in IV-C3, the three controllers used for the compensation with sampling periods T_0 equal to $0.185s$, $0.2s$ and $0.215s$ were designed. The values of the compensated PID controller parameters dealing with each possible jitters are reported in Table I. The

TABLE I. COMPENSATED PID CONTROL PARAMETERS.

T_0	0.185	0.2	0.215
K_p	10.8	9.8	10.5
K_i	0.15	0.12	0.1
K_d	23.85	22	23.5

QoC of the three controllers running at different sampling periods are represented in Figure 8.

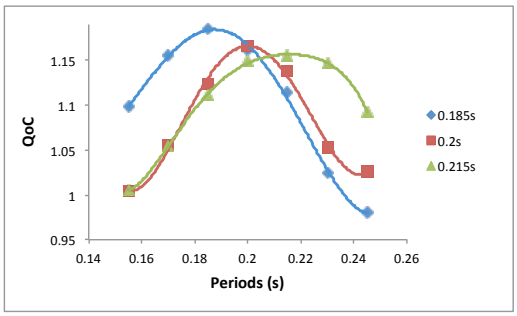


Figure 8. QoC of the three controllers running at different sampling periods.

We compared the system response of PID controllers with and without jitter compensation. Figure 9 shows the jitter degrading effect and the improvement achieved by the compensation. The ideal system response was obtained by running the controller designed with sampling period $T = 0.2s$ without the interference of multitasking. When the PID control task executes at the background of multitasking, the system response of the compensated controller catches more quickly and stays closer to the ideal response with respect to the controller without compensation. Moreover, the system response of the controller without compensation suffers from a larger overshoot and a lower rising speed.

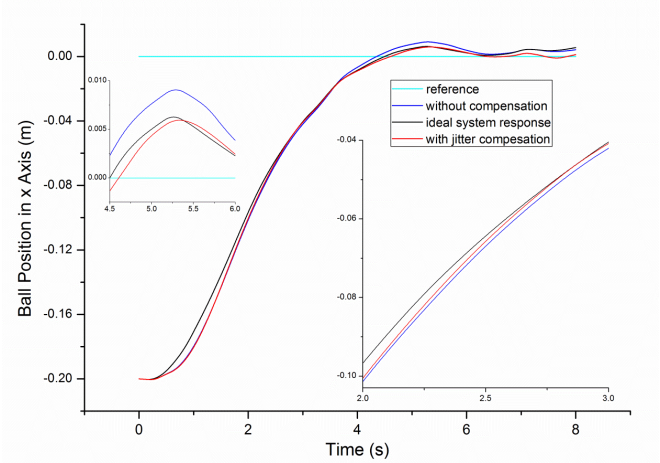


Figure 9. System responses with and without jitter compensation.

B. System testing

Finally, the ball-and-plate system was implemented and tested. The software developed on the two boards runs on

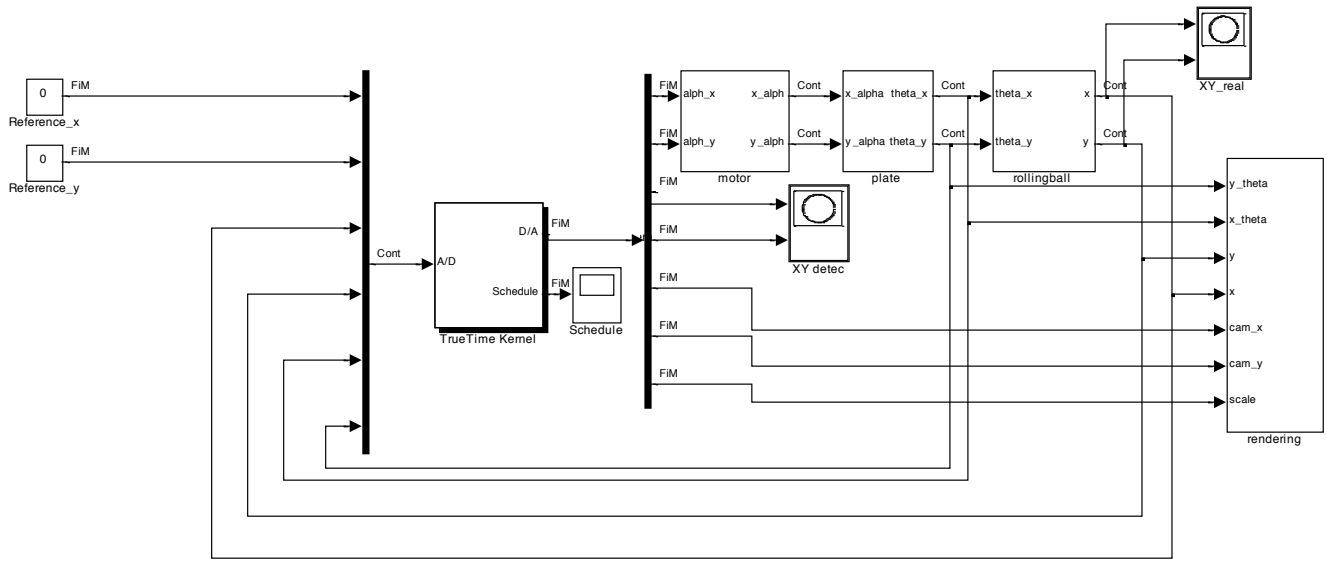


Figure 5. Simulation model.

top of the Erika Enterpise real-time kernel [9]. The desired control performance was obtained by adopting the adaptive mechanisms. Figure 10 illustrates the physical connections and running state of the system.

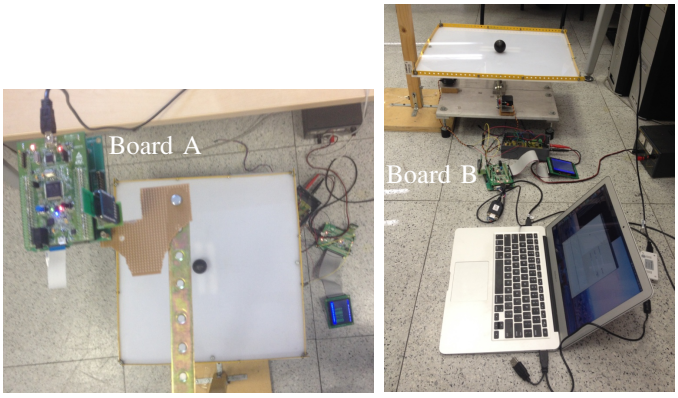


Figure 10. Board A detects the ball position, while Board B controls the motors and communicates with the laptop for monitoring.

VI. Conclusions

This paper illustrated how the usage of adaptive mechanisms in sensing and control activities can improve the system performance in the presence of stringent hardware constraints. To cope with the limitations of computational speed of the embedded board, we presented an adaptive approach that adapts the capture window of the image acquisition process at every execution to improve the detection accuracy. To cope with the performance degradation due to the sampling jitter, jitter compensation techniques adaptively updating the control parameters at each sampling instant proved to be effective in improving control performance. Conditions for finding the set of proper control parameters, a practical issue of applying jitter compensation techniques, are also presented.

As a future work, we plan to investigate the possibility of applying similar adaptive mechanisms to other embedded control applications, especially for those whose system performance is heavily affected by the hardware constraints. Another interesting research direction is to formally prove the correctness of the condition to choose the control parameters for jitter compensation.

References

[1] S. Awtar, et al., “Mechatronic Design of a Ball on Plate Balancing System“, Mechatronics, 12(2), 2002, pp. 217–228.

[2] C. H. Lampert and J. Peters, “Real- time detection of colored objects in multiple camera streams with off-the-shelf hardware components“, Journal of Real-Time Image Processing, 7(1), March 2012, pp. 31–41.

[3] C. E. Lin and C. C. Ker, “Control Implementation of a Magnetic Actuating Ball and Plate System“, International Journal of Applied Electromagnetics and Mechanics, Vol. 27, No. 1-2, 2008, pp. 133-151.

[4] C. C. Ker, C. E. Lin, R. T. Wang, “Tracking and Balance Control of Ball and Plate System“, Journal of the Chinese Institute of Engineering, 30(3), 2007, pp.459-470.

[5] C. H. Lampert and J. Peters, “Image Fuzzy Control on Magnetic Suspension Ball and Plate System“, International Journal of Automation and Control Engineering, 3(2), May 2012, pp. 35-47.

[6] M. A. Moreno-Armendariz, C. A. Perez-Olvera, F. O. Rodriguez, E. Rubio, “Indirect Hierarchical FCMAC Control for the Ball and Plate System“, Neurocomputing, Vol. 73, No. 13-15, 2010, pp. 2454-2463

[7] STMicroelectronics, “http://www.st.com“.

[8] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, K. E. Arzen, “How Does Control Timing Affect Performance? Analysis and Simulation of Timing Using Jitterbug and TrueTime“, IEEE Control Systems Magazine, 23(3): June 2003 pp. 16–30.

[9] Evidence srl Pisa, “http://evidence.eu.co“.

[10] The mathworks,“Simulink User’s Guide“ Nattick, MA USA, 2000.

[11] G. Buttazzo, P. Marti, M. Velasco, “Quality-of-Control Management in Overloaded Real-Time Systems“, IEEE Transactions on Computers, vol. 56, no. 2, February 2007, pp. 253–266.

[12] K.J. Astrom and B. Wittenmark, “Computer-controlled systems“ Third Edition. Prentice-Hall, 1997.

[13] G. Buttazzo, “Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications“ Third Edition, Springer, 2011.

[14] T. Bradski and A. Kaehler “Learning OpenCV“ First Edition O’Reilly 2008.

[15] P. Marti, G. Fohler, K. Ramamritham, J. M. Fuertes, “Jitter Compensation for Real-Time Control Systems“, Proc. 22nd IEEE Real-Time System Symp., Dec. 2001, pp. 39-48.

 ISBN: 978-1-61208-463-3

The Challenge of Transforming State in the Adaptation Objects

Dominic Seiffert

University of Mannheim
Mannheim, Germany
Email: seiffert@informatik.uni-mannheim.de

***Abstract*—When a provided interface and an expected interface need to be connected with each other, this connection is sometimes hindered by signature mismatches. In the world of object-oriented programming where objects play a key role, one important signature mismatch problem occurs when the expected interface expects an object data type that is per se incompatible, although semantically equal, to the object data type delivered by the provided interface. For example, suppose a birthday calendar is the parameter type expected by the expected interface, but another birthday calendar from another developer is the provided parameter type, then a mismatch on object data type occurs. To solve this problem, adaptation is one potential solution. However, because some programming language constructs are not amenable to adaptation, a mechanism based on transformation can be used instead to complement the adaptation process. The challenge is to retrieve the state of the object instance delivered by the provided interface, and to set it to an instance of the object type by the expected interface. In the literature, this problem, however, has been not tackled so far by the object-oriented community. This position paper aims to highlight this challenge and motivate the development of future adaptation tools to solve this problem fully automatically. The challenge is illustrated by typical transformation examples, ranging from more or less trivial to quite challenging tasks.**

Keywords–Signature Mismatches; Object Adaptation; State Transformation

I. INTRODUCTION

Software building blocks [1] can be connected by their provided and expected interfaces in order to create functionality. Unfortunately, direct connection is not always possible because of signature mismatches. A very simple example of a signature mismatch occurs when the provided interface and the interface to connect have different names although they actually provide the same functionality. This problem arises when the vocabulary in the problem domain is different from the vocabulary in the solution domain [2]. Another signature mismatch problem occurs in the object-oriented world where not only primitive data types can be used as parameters and return types but also object types. An important feature of an object is that it holds a state. When an object is defined as a parameter type or return type for a provided interface but the expected interface supports an object type which is per se incompatible, although semantically equal, a solution is required to solve this mismatch. Adaptation provides such a solution because it overcomes signature mismatches [3]. Several approaches have been proposed in the world of object-oriented programming to tackle this problem of signature mismatches such as [4] [5]. However, the problem of matching objects that hold state has not been considered. It is a crucial problem, however, because objects play an important role in object-oriented development. Therefore, it is important to highlight this challenge in order to improve adaptation in the object-oriented world.

The remainder of this paper is structured as follows: Section 2 provides background information on adaptation, especially in the object-oriented world. Section 3 delivers a motivational example to show that this problem is not just an academic one. Section 4 provides an overview of the different problems in more detail. Section 5 provides a conclusion and an outlook on future work.

II. BACKGROUND

In the early 90's Rittri [6], Runciman and Toyn [7] and Cosmo [8] proposed an approach for tackling the problem of retrieving functionality by matching types, laying the basis for the problem of type isomorphism. Two types A and B are definably isomorphic (A $\cong_d$ B) iff there exist functions (λ-terms) M:A $\rightarrow$ B and N:B $\rightarrow$ A such that M $\circ$ N = I_B and N $\circ$ M = I_A, where I_A and I_B are λx : A.x and λx : B.x, the identities of type A and B. The terms are called invertible.

A minor formal explanation is presented in [8] on page 38, which states that any two types A and B are equivalent if we can write two simple transformations h:A $\rightarrow$ B and $h^{-1} : B \rightarrow$ A such that

1) for any function f:A, $h(f) : B$ and $h^{-1}(h(f)) = f$
2) for any function g:B, $h^{-1}(g) : A$ and $h(h^{-1}(g)) = g$.

The main subject of interest are the transformation functions h and h^{-1}, which are written manually by the developers [8][page 38]. Existing approaches, however, have focused on the area of functional languages where objects do not play a key role. Object were first introduced by object-oriented programming [9]. In this context, Pierce defines in [10] on the pages 225 - 228, an object as a data structure that encapsulates some internal state and offers methods to access this state. The internal state is thereby represented by mutable fields that are shared among the methods and are inaccessible from the outside. The latter is a feature named *encapsulation*, i.e., the internal fields should only be accessed from the outside by methods, as first promoted by Parnas [11], by the principle of information hiding. As stated by Cosmo [8] on page 213, in the presence of state, former results on type isomorphism are no longer guaranteed to be complete. Therefore, it is important to tackle this problem in the object-oriented world by adapting superficially incompatible object data types that are equal from the semantic viewpoint to match one another.

 ISBN: 978-1-61208-463-3

The main challenge herein is to retrieve the state from an instance delivered by a provided interface and to set this state to an instance of an object type that can then be delivered to an expected interface. This mechanism will be named *transformation* in the following.

Definition 1: Let *newInstance* be an instance of type *New* that is delivered by a provided interface as a parameter type. Let *oldInstance* be an instance of a type *Old* that is the parameter type by the expected interface. Let *Old* and *New* be per se incompatible, i.e., no instance *newInstance* can be delivered when an instance of *Old* is expected. A transformation is then needed to get the state from the *newInstance*, create a new *oldInstance* and set the state of *oldInstance* appropriately. A transformation is valid when the *oldInstance* is actually what the expected interface expects.

It is important to raise awareness of this problem because automated adaptation in the object-oriented world requires this functionality. For example, one of the most interesting approaches for adaptation in the object-oriented world is based on the idea proposed by Hummel and Aktinson in 2004 [12] to use test cases to find candidates using a component search engine. In the test case, which is a simple unit test case, the client specifies the expected semantics, i.e., the expected interface with provided input and expected output parameters. The test case is used then during the search process by a component search engine in order to validate potential candidates against the semantics specified in the test case. This approach is known as test-driven reuse [4]. During the search, adaptation may become necessary, however, because candidates not necessarily fulfill the expected interface, i.e., signature mismatches may make it impossible to match the expected interface to a provided interface. The implemented adaptation engine tries to overcome signature mismatches using brute-force parameter permutation and so called relaxed signature matching on primitive data types (i.e., when an int is provided and a long is expected, the parameter types are regarded as a valid match [13]). Unfortunately, the approach does not consider the matching of object data types, which is a crucial requirement in object-oriented programming languages. The same applies for other approaches from the same field, proposed by Lemos et al. [14], named Sourcerer, and the search engine S6 proposed by Reiss [15]. The approach proposed by Kell [5] uses mapping rules in order to tackle the problem of object data type matching. This, however, is a cognitive overhead for a developer, especially when the details of the candidate to adapt are not really known. Seiffert and Hummel tackled the problem of matching well known data structures, such as linked lists, stacks and arrays on each other, by providing automated transformation mechanisms [16]. However, these are pre-defined mechanisms and not generally applicable on object types which are unknown beforehand.

As this brief overview reveals, there is currently no existing approach in the object-oriented world that tackles the problem of matching object-data types fully automatically. This will be further motivated in the next section.

III. MOTIVATION

Suppose a client wishes to use a BirthdayCalendar, where it is possible to set and retrieve birthdays for persons as shown by Figure 1. The classes *Person* and *Date* are assumed

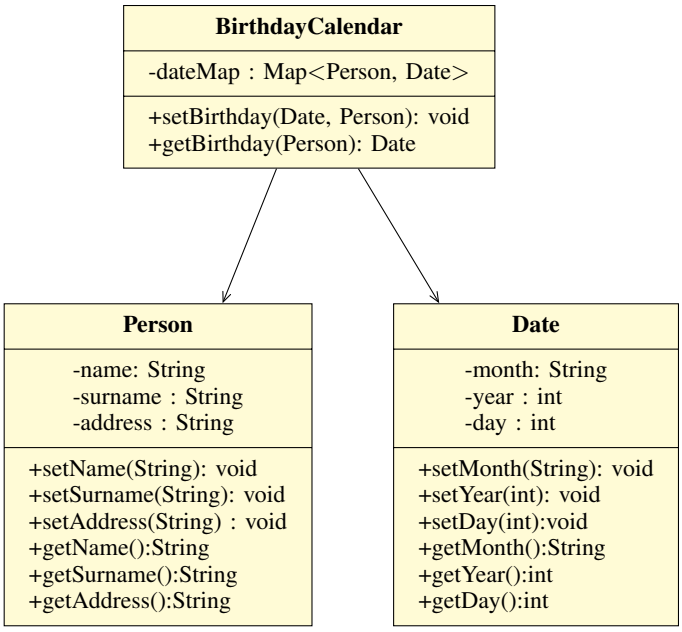


Figure 1. An expected BirthdayCalendar

be already fully implemented and available in the development project. The missing component is the *BirthdayCalendar* which supports the storing and retrieving of birthdays using the *setBirthday* and *getBirthday* methods. In other words, implementations of these methods are still missing. Although this is trivial it provides a simple example to explain the main problem.

Suppose *GeburtstagsKalender* is semantically equivalent to *BirthdayCalendar* (in fact, it is simply an implementation from another vendor for German speaking people). Figure 2 shows the corresponding class diagram. The type “Person” used by the *GeburtstagsKalender* has the same type name as the type Person expected by the *BirthdayCalendar*. The reason, in this case, is simply that both words have the same meaning in English and German. The types Person are semantically equivalent, i.e., a Person used by the *BirthdayCalendar* is described by its “name”, “surname” and its “address”. Correspondingly the type Person used by the *GeburtstagsKalender* is described by the German terms “name”, “nachname” and “adresse”. Neither of the types are connected by a type hierarchy. This is important, because according to the Liskov Substitution Prinicple [17] a subclass can be delivered in any situation where a parent class is expected. The construct *final* prohibits the creation of such a subclass relationship however. This also applies for Datum that is semantically equivalent to Date.

The *GeburtstagsKalender* would be useful because it provides the missing implementation, namely to store birthdays for persons and to retrieve them by the *setBirthday* and *getBirthday* methods. However, since Person and Datum do not match the expected types Person and Date this is not possible.

In order to let the client use *GeburtstagsKalender* to retrieve the expected functionality for storing and retrieving birthdates, the client relies on the well-known object adapter pattern proposed by the “Gang of Four” [18] as illustrated by Figure 3. The idea of this pattern is simple: an adapter implements the expected interface by the client and adapts

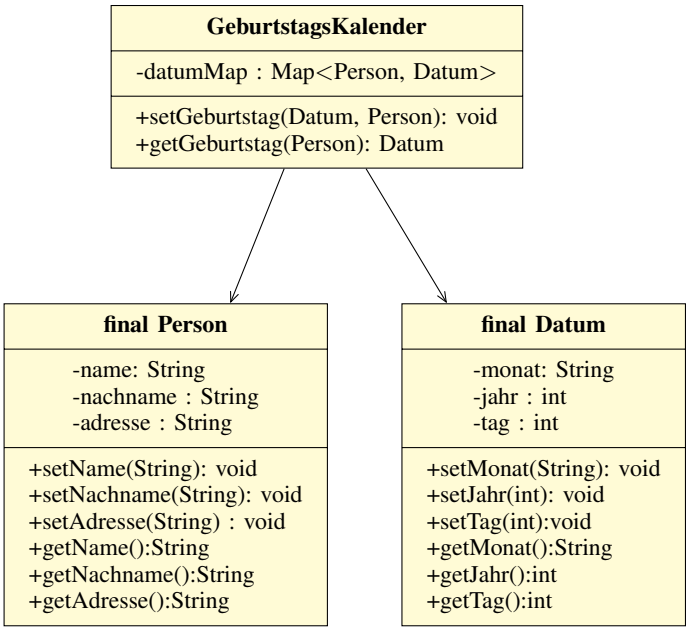


Figure 2. GeburtstagsKalender: a class semantically equivalent to BirthdayCalender

an object with the "wrong" interface. The client can use the adapter as if he would be working with the candidate to adapt directly. That is, all incoming messages are forwarded to the candidate and messages delivered by it are propagated back to the client. If the method names *setBirthday* and *setGeburtstag* would be the only mismatch, it would be sufficient to forward the parameters. However, in this case, mismatches on object data types exist. Therefore, more effort is required by the adapter to provide a suitable *transformation* mechanism.

In a former publication, we clarified the difference between adaptation and transformation [19]. Transformation complements adaptation and becomes relevant when adaptation cannot be applied. In the given example, adaptation would potentially solve the problem of the mismatching parameter types by creating adapters Datum2Date and Person2Person. These adapters are used by the adapter BirthdayCalendar2GeburtstagsKalendar within the *setBirthday* method. The arriving instance of Date would be set then to the Datum2Date adapter by a method *setAdaptee* as the candidate to adapt. This adapter can then be forwarded as a parameter to the *setGeburtstag* method. However, in order to create such an adapter, it must be able to subclass the parameter types Datum and Person of the *GeburtstagsKalender*. This is not possible in this case because of the *final* declaration in the given case for Person and Datum. In this former publication, we explained how transformation can be part of an adapter for building a facade, as a sophisticated example, however, we did not provide a more fine grained distinction for transformations and did not relate the problem on transforming state with type isomorphism. We also did not explain the main difference between objects from the object-oriented world and the abstract data types, used by the web service community in the web service description language, and did not mention the need for applying the transformation mechanism (and therefore also the adaptation mechanism) recursively, as well as the requirements for performing a transformation on an object type.

TABLE I. REQUIRED METHOD MATCHINGS FOR A PERSON TO SUPPORT OBJECT TRANSFORMATION

Output Method (Provided Instance)	Input Method (Expected Instance)
getName	setName
getSurname	setNachname
getAddress	setAddress

In the next section we provide an overview of the different challenges that need to be addressed in order to provide a transformation mechanism to complement adaptation.

IV. Providing a Transformation Mechanism

In order to provide a transformation mechanism, it is necessary to access the state provided by an object instance. Therefore, at least some of the following preconditions need to be fulfilled:

- the state can be accessed through the attributes provided by the object,
- the state can be accessed through the method(s) provided by the object.

According to the information hiding principle originally proposed by Parnas [11] objects should provide access to their internal attributes only through methods that are visible to the outside only. In order to investigate and access the methods on a given type in the Java language the Java Reflection API can be used. In the following, different case are considered, starting from low-level complexity to high-level complexity. The presented problems are enriched with intuitive solutions, but it is up to future research to investigate if these problems can be solved efficiently.

Case 1: In order to transform the state from the Person instance delivered by the *setBirthday* method to an instance of a Person expected by the *setGeburtstag* method, the method matchings shown in Table I have to be realized. The first column names the method of the provided instance whose output parameter needs to be set as an input parameter of the method of the expected instance in the second column.

To transform a Date instance to a Datum instance, the corresponding method matchings are shown in Table II. This challenge of matching output to input values has already been tackled by the web-service community [20], but has not been applied to objects in the context of object-oriented programming. There is a difference between an object and an abstract data type specified as a complex type in a web service description language (WSDL). A complex type does not provide any functionality through methods and WSDL prefers to use the xml schema definition, which provides a set of built-in data types, whereas an object from an object-oriented programming language can be of any type.

Case 2 The given example requires a flat transformation only, i.e., the parameter data types provided by the output methods are primitive data types only. The type of the *address* attribute of the type Person from the *BirthdayCalendar* can be changed to an an object type *Address* that holds data about the address, such as street and city, which can be set and retrieved by corresponding setter and getter methods again.

 ISBN: 978-1-61208-463-3

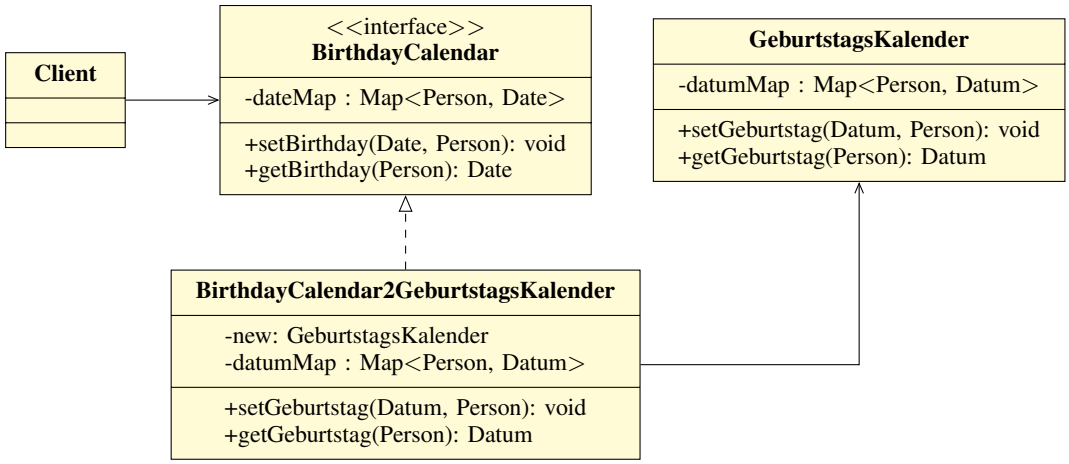


Figure 3. Adapter BirthdayCalendar2GeburtstagsKalender which adapts GeburtstagsKalendar

TABLE II. REQUIRED METHOD MATCHINGS FOR A DATE TO DATUM TO SUPPORT OBJECT TRANSFORMATION

Output Method (Provided Instance)	Input Method (Expected Instance)
getYear	setJahr
getMonth	setMonat
getDay	setTag

The type of *adresse* of the Person type expected by the *GeburtstagsKalender* can be changed to an object type *Adresse* accordingly, i.e., to an object type that is semantically equal but superficially incompatible. The problem of state transformation then needs to be applied recursively.

Case 3 Suppose the Person type of the *BirthdayCalendar* does not manage all its data by means of single fields, but by an internal array where each position specifies the content. For example, the position 0 might specify the name, position 1 the surname and position 2 the address, where for address the primitive data type, String, is assumed again. This array can be set by a *setInformation(String[] data)* method and retrieved by a *getInformation():String[]* method accordingly. In such a case, to provide a transformation mechanism, the array instance needs to be retrieved by the *getInformation* method and all possible permutations need to be applied as method invocation on the methods *setJahr*, *setMonat* and *setTag*, to set the content from the array to an instance of type Person appropriately as expected by the *GeburtstagsKalender*.

Case 4: Suppose the opposite situation to that stated in Case 3 occurs, i.e., suppose the Person type of *GeburtstagsKalender* expects an array of values describing the person, and suppose the Person type of the *BirthdayCalendar* uses single fields instead as in the initial example. Then, the adapter performing the transformation mechanism needs to create a new array instance which is then filled by invoking the getter methods on the delivered Person instance. This again requires that all possibilities are permutated in the worst case.

V. CONCLUSION

The aim of this position paper is to highlight the problem of transforming the state of object instances. This problem occurs when a provided interface delivers an object type, but the expected interface expects an object instance that is superficially incompatible with the type of the delivered instance and an adaptation mechanism itself can not be applied. In this paper we have used two implementations of birthday calendars from different developers that are are not connected by a type hierarchy, but provide the same semantics, to illustrate the problem. These types have the same functionality, i.e., the only differences are syntactic. Even semantic differences do not necessarily stop a transformation mechanism from being applied because transforming the state from a provided instance to an expected instance may be enough to let the expected interface use it. For example, a queue and a stack only share similar semantics. Equivalence is attained, however, when the queue instance created by transformation satisfies the expected interface. For this transformation, elements are retrieved from the stack and set in a queue instance which is forwarded. In the optimal case, this should be performed fully automatically by a transformation mechanism provided by the adapter. This problem of state transformation is, to the best of the author's knowledge, neglected in the literature so far. However, it needs to be tackled because object types provide an important signature mismatch problem to be solved. The availability of mechanisms to solve this problem fully automatically could, for example, significantly increase the recall of code search engines. Therefore, more tools and approaches need to be developed to automatically solve this problem.

REFERENCES

[1] M. Lenz, H. A. Schmid, and P. F. Wolf, "Software reuse through building blocks," *IEEE Software*, vol. 4, no. 4, pp. 34–42, July 1987.

[2] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, "The vocabulary problem in human-system communication," *Communications of the ACM*, vol. 30, no. 11, pp. 964–971, November 1987, DOI: 10.1145/32206.32212.

[3] S. Becker, A. Brogi, I. Gorton, S. Overhage, A. Romanovsky, and M. Tivoli, "Towards an engineering approach to component adaptation," in *Architecting Systems with Trustworthy Components*, ser. Lecture Notes in Computer Science, R. Reussner, J. A. Stafford, and C. Szyperski, Eds. Springer Berlin Heidelberg, 2006, vol. 3938, pp. 193–215, DOI: 10.1007/11786160_11.

[4] O. Hummel and W. Janjic, "Test-driven reuse: Key to improving precision of search engines for software reuse," in *Finding Source Code on the Web for Remix and Reuse*, S. Sim and R. Gallardo-Valencia, Eds.

 ISBN: 978-1-61208-463-3

Springer New York, 2013, pp. 227–250, DOI: 10.1007/978146146596-6_12.

[5] S. Kell, "Component adaptation and assembly using interface relations," in *Proceedings of the ACM international conference on Object oriented programming systems languages and application*, ser. OOPSLA'10, vol. 45, no. 10. New York, NY, USA: ACM, 2010, pp. 322–340, DOI: 10.1145/1869459.1869487.

[6] M. Rittri, "Using types as search keys in function libraries," *Journal of Functional Programming*, vol. 1, no. 1, pp. 71–89, 1991.

[7] C. Runciman and I. Toyn, "Retrieving re-usable software components by polymorphic type," *Journal of Functional Programming*, vol. 1, no. 2, pp. 191–211, 1991.

[8] R. D. Cosmo, *Isomorphisms of Types: from lambda calculus to information retrieval and language design*, R. V. Book, Ed. Birkhäuser, 1995.

[9] G. Booch, "Object-oriented development," *IEEE Transactions on Software Engineering*, vol. SE-12, no. 2, pp. 211–221, 1986.

[10] B. C. Pierce, *Types and Programming Languages*. The MIT Press, 2002, ISBN: 978-0262162098.

[11] D. L. Parnas, "On the criteria to be used in decomposing systems into modules," in *Communications of the ACM*, vol. 15, no. 12. ACM, 1972, pp. 1053–1058, DOI: 10.1145/361598.361623.

[12] O. Hummel and C. Atkinson, "Extreme harvesting: Test driven discovery and reuse of software components," in *Proceedings of the International Conference on Information Reuse and Integration*, ser. IEEE-IRI, 2004, pp. 66 – 72.

[13] ——, "Automated creation and assessment of component adapters with test cases," in *Component-Based Software Engineering*, ser. Lecture Notes in Computer Science, L. Grunske, R. Reussner, and F. Plasil, Eds., vol. 6092. Springer Berlin Heidelberg, 2010, pp. 166–181, DOI: 10.1007/9783642132384_10.

[14] O. A. L. Lemos, S. Bajracharya, J. Ossher, P. C. Masiero, and C. Lopes, "A test-driven approach to code search and its application to the reuse of auxiliary functionality," *Information and Software Technology*, vol. 53, no. 4, pp. 294–306, April 2011, DOI: 10.1016/j.infsof.2010.11.009.

[15] S. P. Reiss, "Semantics-based code search," in *Proceedings of the 31st International Conference on Software Engineering (ICSE 2009)*. IEEE Computer Society, 2009, pp. 243 – 253, DOI: 10.1109/ICSE.2009.5070525.

[16] D. Seiffert and O. Hummel, "Adapting arrays and collections: Another step towards the automated adaptation of object ensembles," in *Lecture Notes in Computer Science, ICSR 2015*, ser. Lecture Notes in Computer Science, I. Schaefer and I. Stamelos, Eds., vol. 8919. Springer International Publishing Switzerland, 2015, pp. 348 – 363.

[17] B. H. Liskov and J. M. Wing, "A behavioral notion of subtyping," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 16, no. 6, pp. 1811–1841, 1994.

[18] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Pearson Education, 1994, ISBN: 9780321700698.

[19] D. Seiffert and O. Hummel, "How adaptation and transformation complement each other to potentially overcome signature mimismatch on object data types on the basis of test-cases," in *Adaptive 2015: The Seventh International Conference on Adaptive and Self-Adaptive Systems and Applications*. IARA, 2015, pp. 98–102, ISBN: 9781-612083919.

[20] H. R. Motahari Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casati, "Semi-automated adaptation of service interactions," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 993–1002.

 ISBN: 978-1-61208-463-3

Ontology-based Automatic Adaptation of Component Interfaces in Dynamic Adaptive Systems

Yong Wang, Dirk Herrling, Peter Stroganov, and Andreas Rausch
Department of Informatics
Technical University Clausthal
Clausthal-Zellerfeld, Germany
e-mail: yong.wang, dirk.herrling, peter.stroganov, andreas.rausch@tu-clausthal.de

***Abstract*—Dynamic adaptive systems are systems that change their behavior at runtime. Behavioral changes can be caused by user's needs, or based on context information if the system environment changes. The Dynamic Adaptive System Infrastructure (DAiSI) has been developed as a platform for such systems. It is a run-time infrastructure that operates on components that comply to a DAiSI-specific component model. DAiSI-based systems are "open" by design. The run-time infrastructure can integrate components into the system that were not known at design-time. To control the system configuration of such an open and self-organizing system, a configuration service has been developed that can make use of application blueprints to ensure application architecture conformance. Components in a DAiSI system communicate with each other through services. Services are described by domain interfaces, which have to be specified by the component developer. Components can utilize services, provided by other components as long as the respective required and provided interfaces are compatible. However, sometimes services seem to be doing the same thing, e.g., provide the same data or operations, but differ on a syntactical level. Therefore, in this article, we present an approach which enables the use of syntactically incompatible services. We developed an ontology-based method for the generation of an adapter that connects services, which provide the right data in the wrong format. In this paper we present a method to describe interfaces of components and an algorithm to automatically generate adapters between them.**

Keywords—component models; self-adaptation; dynamic adaptive systems; ontology.

I. INTRODUCTION

An increasing interest in dynamic adaptive systems could be observed in the last two decades. A platform for such systems has been developed in our research group for more than ten years. It is called Dynamic Adaptive System Infrastructure (DAiSI). DAiSI is a component based platform that can be used in self-managed systems. Components can be integrated into or removed from a dynamic adaptive system at run-time without causing a complete application to fail. To meet this requirement, each component can react to changes in its environment and adapt its behavior accordingly.

Components are developed with a specific use-case in mind. Thus, the domain interfaces describing the provided and required services tend to be tailored to a very specific application. This effect limits the re-use of existing components in new applications. One measure to minimize redeveloping existing components is to increase reusability. The reuse of existing components is one key aspect in software engineering. However, re-using components in other application contexts than they have been originally developed for is still a big challenge. This challenge gets even bigger, if such components should be integrated into dynamic adaptive systems at run-time.

A valid approach to tackle this challenge is adaptation. Because of the dynamic adaptive nature of DAiSI applications, DAiSI components are considered as black boxes. Their capabilities and behavior are specified by interfaces that describe required– and provided services. In this approach, we suggest a solution to couple provided and required services that are syntactically incompatible. On a semantical level, the provided service does offer the needed data or operations. To be able to utilize a specific provided service, we suggest to construct an adapter that enables interoperability between services that are only compatible on some semantical level.

The goal of an adapter is to enable communication between two formerly incompatible components. In order to translate different representations of data, a common knowledge-base is needed. In this work we use a central ontology as the common knowledge-base. To illustrate that this approach is suitable for adaptive systems, we extend our DAiSI infrastructure by an ontology-based adapter engine for service adaptation. To strengthen the dynamic adaptive nature of the DAiSI, we generate these adapters at run-time. We argue that these adapters cannot be generated at compile time, as the different components that should interact with each other are not known at compile time, but only at integration time, which is the same as run-time in dynamic adaptive systems.

The rest of this paper is structured as follows: In Section II, we describe the already sketched problem in more detail. Section III gives an overview of relevant related work. In Section IV, we give a short overview of the DAiSI component model and a few hints for further reading. Section V explains, how the adaptation of services with the help of an ontology works. In Section VI, the algorithm for the adapter generation is shown in more detail, before the paper is wrapped up by a short conclusion in Section VII.

 ISBN: 978-1-61208-463-3

II. PROBLEM DESCRIPTION

Whenever a dynamic adaptive application is developed, the interfaces between the components are specified at an early stage. They are very domain specific and their definition is driven by the use cases of the future application in mind. On the other hand, many applications run in a shared context with other applications from different domains.

Harmonizing one large interface pool among different developers from different vendors that operate in different domains is a tedious task, which often results in a slow standardization process. This slows the development process down and, especially in dynamic adaptive systems, diminishes the chances for the development of new applications. Developers will in those cases often start their own interface pool. This, on the other hand, reduces the chances to re-use existing components from other domains.

Additionally, the management of one central interface pool in a distributed system does not scale well. One way to mitigate this issue would be a de-centralization. To tackle these challenges, we propose to keep the domain interfaces un-harmonized. To be able to use a service across domains, we propose to adapt syntactically incompatible services by on-the-fly generated adapters. To be able to do so, we require every interface pool to use an ontology. By either merging these ontologies later on, or by using distributed ontologies we ensure that interfaces from different interface pools share a common semantic.

Components offer provided services. To be able to do so, they may require others and thus, specify required services. Provided and required services stand in relation to each other, mapping which required services are necessary to produce which provided services. In the graphical notation for DAiSI components provided services are marked as filled circles, required services are noted as semi-circles (similar to the UML lollipop notation [13]) and the relation between those two are marked as bars across the component, linking provided and required services (cf. Figure 1).

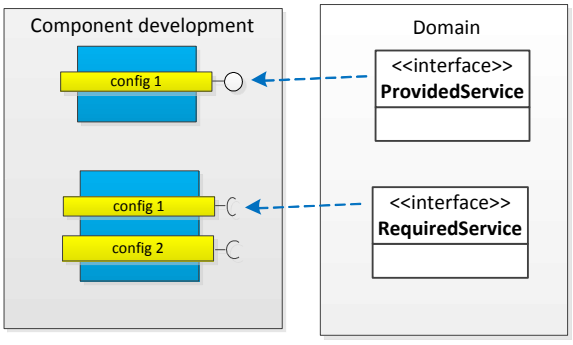


Figure 1. DAiSI components and domain-specific interface definitions.

We propose that services that are semantically compatible, but lack compatibility on a syntactical level, should be usable. We suggest to generate adapters between the different services and define three different adaptation scenarios to face the following three types of incompatibility: Different Naming, Different Data Structure, and Different Control Structure. We believe that we can connect all semantically compatible but syntactically different services using these three types of adapters.

A. Different Naming

By "Different Naming" we denote cases in which the names of interfaces describing services or names of functions do not match. While they are syntactically different, their names share the same semantics and could be used synonymous. The first example, depicted in Figure 2, shows two interfaces: `PowerInfo` and `PowerQuality`. They are named differently, but offer the same functionality. Each of them defines one of the following methods: `update`, and `save` respectively. The names of their parameters are identical and so are the return types.

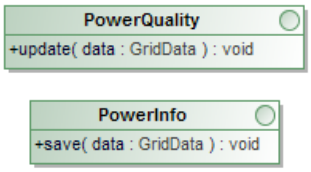


Figure 2. Example of two interfaces with Different Naming.

B. Different Data Structure

In this type of conflict, the names of the interfaces and their functions are the same. However, the parameters differ in their data types. The encapsulated data however is similar and the data structures can be mapped to each other. In Figure 3 in the Different Data Structure example an interface `PowerQuality1` is depicted. It contains a function `saveGridInfo` which processes a parameter of the type `GridData`. In the interface `PowerInfo1`, there are two other functions with the same name but with two different input parameters.

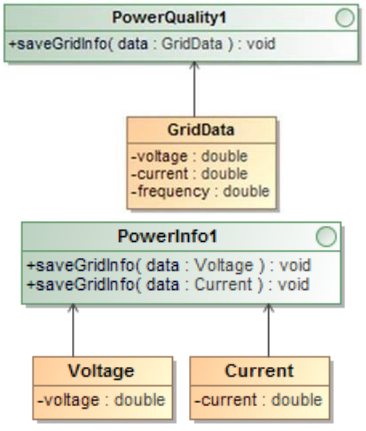


Figure 3. Example of two interfaces with a Different Data Structure.

C. Different Control Structure

In this case, the functions of two different interfaces can

not be mapped directly to each other. To obtain valid results, the control structure has to be modified. In the example in Figure 4, two interfaces `UserInterface` and `UserManager` are given. By definition a username should be composed of the first– and the last name of a person. As such, the two functions `getFirstName` and `getLastName` from the `UserManager` interface in comparison provide the same information as `getUsername` from the `UserInterface` interface.

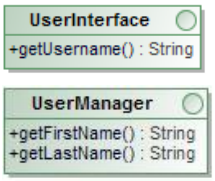


Figure 4. Example of two interface requiring Different Control Structures.

To enable the mapping between interfaces, a common knowledge-base is needed. Because of the issues stated earlier, it should not be mandatory that both sets of interface definitions are of the same domain. A common knowledge-base defined by ontologies can be generated using merging or other integration mechanisms on classical ontology languages or by using a distributed ontology language. Both interfaces do not need to contain information on how to interpret the data of each other. That means that interfaces can be developed independently, without knowing anything about a possible re-use in another system.

III. Related Work

A dynamic adaptive system is a system that adjusts its structure and behavior according to the user's needs or to a change in its system context at run-time. The DAiSI is one example of an infrastructure for dynamic adaptive systems [4][7][15][17]. It has been developed over more than a decade by a number of researchers. This work is based on DAiSI and extends the current run-time infrastructure.

According to a publication of M. Yellin and R. Storm, challenges regarding behavioral differences of components have been tackled by many researchers [22]. The behavior of the interface of a component can be described by a protocol with the help of state-machines. The states of two involved components are stored and managed by an adapter. In further steps of this method, an ontology is used as a language library to describe a component's behavior. To automate the adaptation of services, a semi-automated method has been developed to generate adapters with the analysis of a possible behavioral mismatch [5][20].

Another solution for the connection of semantically incompatible services is presented in [5]. They used buffers for the asynchronous communication between services and translated the contents of those buffers to match the syntactical representations of the involved services. The behavioral protocols of services can automatically be generated with a tool that is based on synthesis– and testing techniques [18]. Ontologies are used in their method to describe the behavior of components and to create a tool for automated adaptation [8]. However, some components require a very complex state-machine; the development of which can easily become very expensive. Thus, in this work, we present another way that does not rely on the consideration of dependencies within the behavior or the involved interfaces.

The method of transformation of an ontology into interfaces is already integrated into Corba Ontolingua [11]. With this tool an ontology can be transformed into the interface definition language (IDL). A. Kalyanpur [21] has developed a method which allows automatic mapping from Web Ontotolgy Language (OWL) to Java. The Object Management Group (OMG) [13] has defined how to transform the Unified Markup Language (UML) into an ontology. With their method, UML classes are first converted into a helper class and then transformed into an ontology [19]. G. Söldner [12] has shown how to transform the UML itself into an ontology. A downside of the above methods: The interface and the ontology have a strong relation. If a developer changes the ontology, all interfaces which are linked to this ontology have to be modified. In this work, we decoupled this strong relation. Alternating a part of the ontology now only affects the interfaces directly linked to it.

Matching and merging existing ontologies is still a big challenge regarding its speed and accuracy. To simplify this, many application interfaces (APIs) have been developed, e.g., Agreement Maker [9] and Blooms [14]. Most of them follow a survey approach [10], or use data available on the Internet [6]. Many methods are used to match entities to determine an alignment, like testing, heuristics, etc. To improve accuracy, many of them use third-party vocabularies such as WordNet or Wikipedia. However, ontology merging is simply used in our approach and we did not conduct further research on the challenges mentioned.

IV. The DAiSI Component Model and Infrastructure Services

The DAiSI component model can best be explained with a sketch of a DAiSI component. Figure 5 shows a DAiSI component. The blue rectangle in the background represents the component itself. The provided and required services are depicted with full– and semi circles, as stated earlier. The dependencies between these two kinds of services are depicted by the yellow bars. They are called component configurations. At run-time, only one component configuration can be active. Being active means that all connected, required services are present and consumed (the dependencies could be resolved), and the provided services are being produced. To avoid conflicts the component configurations are sorted by quality with the best component configuration noted at the top (`Conf1` in Figure 5) and the least favorable one noted at the bottom (`Conf2` in our example). The following paragraphs explain the DAiSI component model, depicted in Figure 6. The component model is the core of DAiSI and has been covered in much more detail in [2]. The component configurations (yellow bars) are represented by

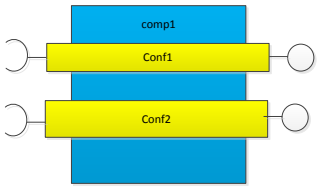


Figure 5. A DAiSI component.

the class of the same name. It is associated to a `RequiredServiceReferenceSet`, which is called a set to account for cardinalities in required services. The provided services are represented by the `ProvidedService` class. Interface roles, represented by `InterfaceRole`, allow the specification of additional constraints for the compatibility of interfaces that use run-time information, bound services and the internal state of a component, and are covered in more detail in [1].

To be able to narrow the structure of a dynamic adaptive system down, blueprints of possible system configurations can be specified. The classes `Application`, `Template`, `RequiredTemplateInterface`, and `ProvidedTemplateInterface` are the building blocks in the component model that are used to realize application architecture conformance. One `Application` contains a number of `Templates`, each specifying a part of the possible application. A `Template` defines (needs and offers) `RequiredTemplateInterfaces` and `ProvidedTemplateInterfaces` which refer to `DomainInterfaces` and thus form a structure which can be filled with actual services and components by the infrastructure. More details about templates and application architecture conformance in the DAiSI can be found in [2].

The DAiSI infrastructure is composed of the DAiSI component model, a registration service, which works like a directory for running DAiSI components, and a configuration service which manages how provided– and required services are connected to each other and what component configurations are marked as active. The configuration service constantly checks (either periodically, or event-driven), if the current system configuration (active component configurations, component bindings, etc.)) can be improved. For the adaptation of syntactical incompatible services, we added a new infrastructure service: The adapter engine. The adapter engine keeps track of all provided and required services in the system. Whenever a new DAiSI component enters the system, the adapter engine analyzes its provided services and generates adapter components (which are DAiSI components themselves) to all syntactically incompatible, but semantically compatible services. We will describe this process in the following in more detail.

Figure 7 shows the structure of the adapter engine. It computes on the basis of descriptions of services (provided and required) and generates adapter components. The information collector aggregates the information of provided– and candidates for required services (e.g., methods, parameters, and return types). The mapper component compares the gathered information and computes an assignment list, which maps the information from provided services to candidates for required services, the provided service could satisfy. The generator takes the assignment lists from the mapper and spawns new DAiSI components which each could map one provided service to a semantically compatible required service. The adapter engine keeps track of the lifecycle of every DAiSI component. Whenever a DAiSI component leaves the system, the adapter engine destroys all generated adapters and thereby removes them from the system.

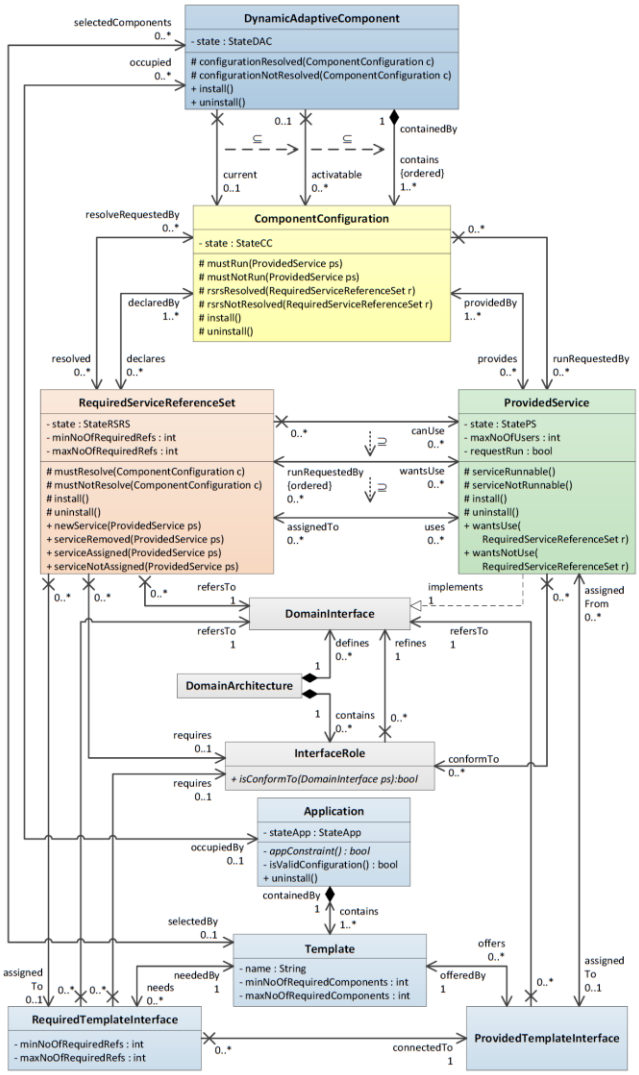


Figure 6. The DAiSI component model.

Figure 8 shows the process and the involved DAiSI components. The component `comp.a` enters the system and provides the service `B`. The adapter engine analyzes the service `B` and, together with its previously built knowledgebase, comes to the conclusion that `comp.b` could use service `B`, but they are syntactically incompatible. The adapter engine can not find another candidate to use service `B`. Thus, it generates only one adapter – a DAiSI component called adapter. It requires the service `B` and provides service `A`. The

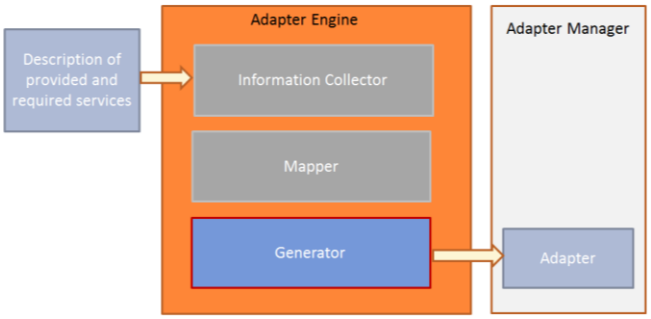


Figure 7. Structure of the adapter engine.

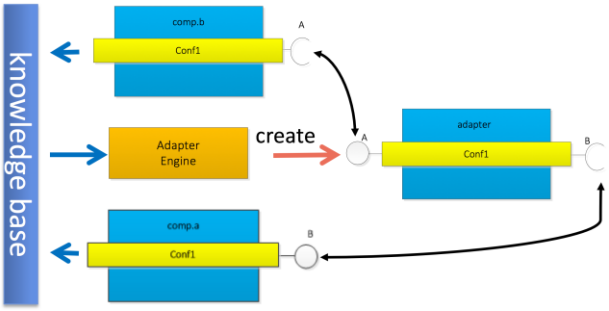


Figure 8. Adaptation process with adapter engine.

DAiSI configuration service binds `comp.b` to adapter and adapter to `comp.a`. The dependency of `comp.b` could be resolved.

V. Interface Description of DAiSI Components with Ontologies

A machine-readable interface description that includes the important semantical information is a key aspect of our concept. Fortunately, making semantic information machine readable, is a well researched and understood field of computer science. For our system, we use a three-layer ontology structure for the construction of the knowledge-base. The upper layer is called UpperOntology layer. In this layer, basic knowledge is defined. Such knowledge can be divided in different upper ontologies. If need be, e.g., if two dynamic adaptive system instances are being merged, the corresponding upper ontologies can be merged. Merging ontologies is a different research area on which we do not focus, however the available results are sufficient for our work.

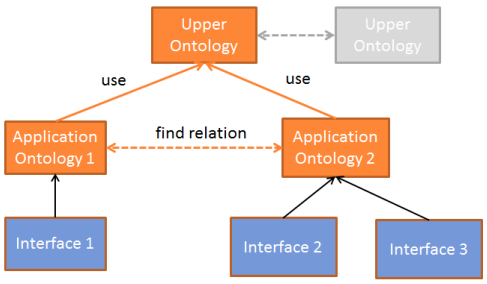


Figure 9. Three-layer ontology structure.

Figure 9 shows the three-layer ontology structure, we used. Every application or domain defines its own upper ontolgy. In the application layer of the ontology, which is the second, or middle layer, all necessary definitions can be found that are relevant for an application. The interface layer of the ontology is the lowest level. It represents the domain interfaces, more precisely their names, methods, parameters and return types. The code of the domain interfaces is directly connected to the ontology. This structure of a three-layer ontology has the main advantage that every part can be developed separately. Every fragment of a layer can be merged with other fragments using ontolgy-merging and ontolgy-mapping.

Figure 10 shows the layout of the ontology for the application example presented in the beginning of this paper. We used two upmost ontologies – Upper Ontology and UML Schema Ontolgy. All definitions and relations for the interfaces, like methods, parameters, or return types can be found in these two ontologies. In the application layer, the ontology data is split by topics. Every information in any of the ontologies can be used in any interface. Datatype classes can also be defined directly in the upper ontology. The following examples show, how the Ontology is defined.

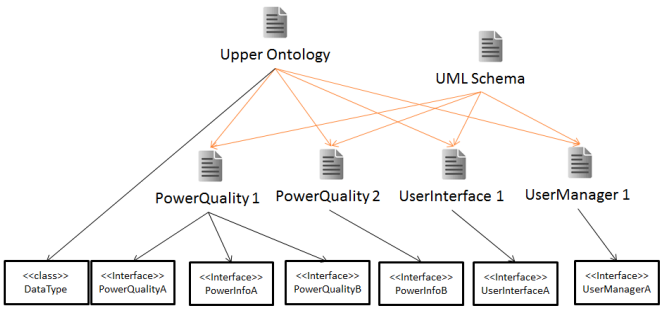


Figure 10. Data structure of the example.

A. Upper Ontology and UML Schema

Figures Figure 11 Figure 12 show graphic representations of the ontologies Upper Ontology and UML Schema.

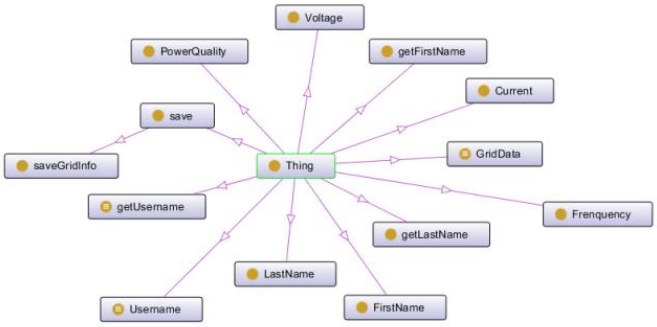


Figure 11. Upper ontology.

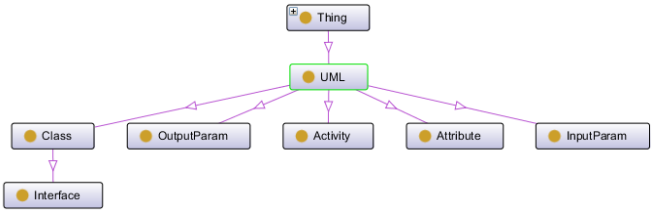


Figure 12. Ontology for UML-Schema.

B. Application Ontology

The ontology file for the interface layer describes all definitions for elements that are used in one application. Every element in an interface, e.g., the interface name, the names of methods, input– and output parameters, are defined as one individual in the ontology. Relations between the elements of different interfaces are also defined in this ontology.

1) Interface name description

Consider the interface `UserManager` which is defined in the ontology of the same name. It is an individual of the upper ontology. Its type is defined as `ont:Interface`, which again is defined in the ontology for the UML-Schema. The code-snippet in Figure 13 shows the definition of the `UserManager` in OWL.

```
<!-- …/ont.owl#UserManager -->

<owl:NamedIndividual
   rdf:about="&ont;UserManager">
   <rdf:type rdf:resource="&ont;Interface"/>
</owl:NamedIndividual>
```

Figure 13. Example of an interface name in the application ontology.

2) Function name description

A function name is also defined as an individual and is of the type `ont:Activity`. To define the relation of the function to the output parameters the individual `ont:hasOutParam` is used. The code-snippet in Figure 14 shows the function `getLastName` as an example; it defines `LastName` as an output.

```
<!-- …/ont.owl#getLastName -->

<owl:NamedIndividual
   rdf:about="&ont;getLastName">
   <rdf:type rdf:resource="&ont;Activity"/>
   <rdf:type rdf:resource="&ont;getLastName"/>
   <ont:hasOuputParam
      rdf:resource="&ont;LastName"/>
</owl:NamedIndividual>
```

Figure 14. Example for the description of a function name in ontology.

3) Input and Output Parameter

Input and output parameters provide important information for the adaptation. Parameter types have to be defined exactly like input and output parameters. The code-snippet in Figure 15 shows the definition for `FirstName`.

```
<!-- …/ont.owl#FirstName -->

<owl:NamedIndividual  rdf:about="&ont;FirstName">
   <rdf:type rdf:resource="&ont;OutputParam"/>
   <rdf:type rdf:resource="&ont;FirstName"/>
</owl:NamedIndividual>
```

Figure 15. Example definition of an output parameter.

C. Java-Annotations for the Interface Description

Our prototype is implemented in Java. We use an aspect oriented method – annotations in Java as a link between the ontology and the actual implementation. In an interface, every element has at least one label that links it to the ontology. Every label has an attribute `hasName` to reference the ontology. Ontology names can be found in the application layer. Interface names, for example, need only one label: `@Interfacename`. Functions have three types of labels: `@Activity`, `@OutputParam` and `@InputParameter`. The label for input or output is used only if a function has input– or output parameters. With the help of annotations, the definition of elements of an interface is decoupled from the actual ontology. This measure was taken to ease the changes of either an interface or the ontology, without the necessity to alter both. The code-snippets in Figure 16 present two Java interfaces as examples.

```
@Interfacename(hasName = "PowerQuality1")
public interface PowerQuality {
   @Activity(hasName = "save")
   public void save(
      @Inputparam(hasName = "GridData")
         GridData griddata);
   }
```

```
@Interfacename(hasName = "UserInterface1")
public interface UserInterface1 {
   @Activity(hasName= "getUsername")
   @OutputParam(hasName= "username")
   public String getUsername();
}
```

Figure 16. Two example interfaces with annotations.

VI. Algorithm for Adapter Generation

In this section we describe the basic concept of the adapter generation in Java, the process for interface comparison, and the inner workings of the generated adapters. Later-on, examples of adapter actions will be shown.

A. Basic principle of the adapter

Every adapter is a DAiSI component, connecting two different interfaces. Figure 17 shows `comp.C` as an example adapter component. The implementation in Java translates a function call from one (`update`) to another (`save`). The provided service of the adapter implements the required interface. The mapping between the required and provided interfaces is implemented in functions of the required interface.

 ISBN: 978-1-61208-463-3

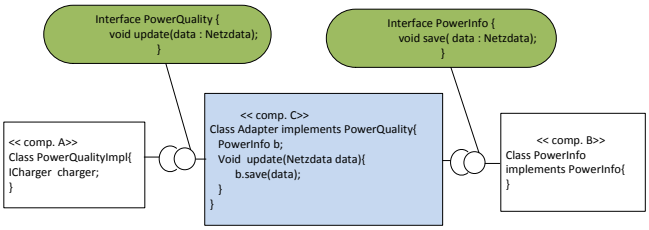


Figure 17. Basic principle of the adapter between two interfaces.

B. *Process for interface comparison*

The information collector in the adapter engine collects instances of the annotations in the Java interface definitions. The mapper uses this collected information to search all dependent instances in the ontology, which are stored in the knowledge-base. The mapper searches for all required interfaces which could possibly use the provided interface. Required services can use provided services, if they are semantically compatible. This path, from required interfaces to provided interfaces is used as a mapping to create the adapter components. Figure 18 shows schematically how the adapter generation works.

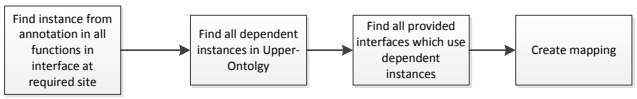


Figure 18. Process for mapping required interfaces to provided interfaces.

C. *Scenario examples*

1) *Different names*

In this case, two functions in required and provided interfaces offer the same functionality, but are named differently. The ontology is used to find the relationship between two functions. The adapter engine generates an adapter component. The required service of the adapter component calls the method of the provided service. Figure 19 shows a UML activity diagram of the behavior of the generated adapter.

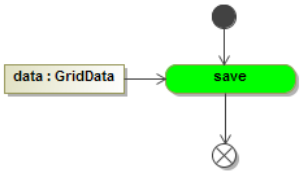


Figure 19. Activity diagram for the call of the `update` method. The call is adapted to the `save`-method.

2) *Different Data Structure*

In the second example, the data structures of parameters are different. For the mapping between parameters, a mapping scheme is searched in the ontology. The adaptation component calls the function from the provided interface using the found mapping for the parameters. Figure 20 shows an UML activity diagram of the behavior of the generated adapter.

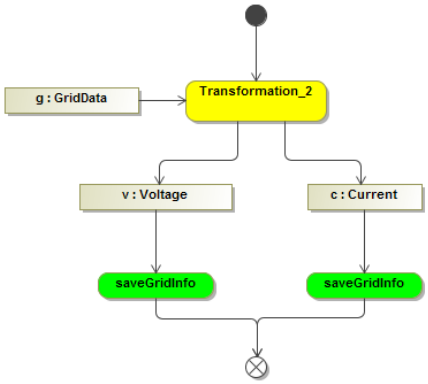


Figure 20. Activity diagram for the adaptation of different data structures

3) *Different Control Structure*

Composition is used, if one function of a required interface can be composed into two or more functions of a provided interface. In this case, the adaptation component calls all functions whose return values can be composed to the required data and composes their return values to match the return value of the adapted interface. Figure 21 shows how two functions are called to account for a difference in control structures.

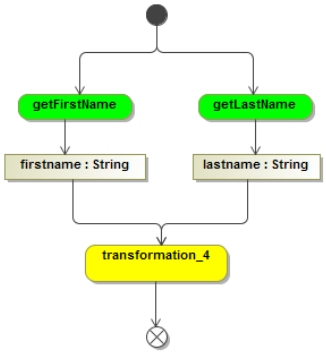


Figure 21. Activity diagram for the adaptation of different control structures

VII. CONCLUSION

In this work, we presented the newest enhancement to the DAiSI: A new infrastructure service. The adapter engine is prototypically implemented with Java and OWL API [3]. It allows the binding of syntactically incompatible services with the help of generated adapters. One of the main benefits is a possible increase of re-use of components across different domains. The layered structure of ontologies allows a collaborative, distributed development.

VIII. FUTURE WORK

In further steps, we will use a distributed ontology, so that every component can be linked directly to the ontology, describing its structure.

 ISBN: 978-1-61208-463-3

REFERENCES

[1] H. Klus, D. Herrling, and A. Rausch, "Interface Roles for Dynamic Adaptive Systems", in Proceeding of ADAPTIVE 2015, The Seventh International conference on Adaptive and Self-Adaptive Systems and Applications, 2015, pp. 80–84.

[2] H. Klus, A. Rausch, and D. Herrling, "Component Templates and Service Applications Specifications to Control Dynamic Adaptive System Configuration", in Proceedings of AMBIENT 2015, The Fifth International Conference on Ambient Computing, Applications, Services and Technologies, Nice, France, 2015, pp. 42–51.

[3] The OWL API, https://github.com/owlcs/owlapi/wiki, [Online], December 2015, retrieved: 02.2016.

[4] H. Klus and A. Rausch, "DAiSI–A Component Model and Decentralized Configuration Mechanism for Dynamic Adaptive Systems", in Proceedings of ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications,Venice, Italy, 2014, pp. 595–608.

[5] C. Canal and G. Salaün, "Adaptation of Asynchronously Communicating Software", in Lecture Notes in Computer Science, vol. 8831, 2014, pp. 437–444.

[6] M. K. Bergmann, "50 Ontology Mapping and Alignment Tools", in Adaptive Information, Adaptive Innovation, Adaptive Infrastructure, http://www.mkbergman.com/1769/50-ontology-mapping-and-alignment-tools/, July 2014, [Online], retrieved: 02.2016.

[7] H. Klus, "Anwendungsarchitektur-konforme Konfiguration selbstorganisierender Softwaresysteme", (Application architecture conform configuration of self-organizing software-systems), Clausthal-Zellerfeld, Technische Universität Clausthal, Department of Informatics, Dissertation, 2013.

[8] A. Bennaceur, C. Chilton, M. Isberner, and B. Jonsson, "Automated Mediator Synthesis: Combining Behavioural and Ontological Reasoning", Software Engineering and Formal Methods, SEFM – 11th International Conference on Software Engineering and Formal Methods, 2013, Madrid, Spain, pp. 274–288.

[9] D. Faria, C. Pesquita, E. Santos, M. Palmonari, F. Cruz, and M. F. Couto, The AgreementMakerLight ontology matching system, in On the Move to Meaningful Internet Systems: OTM 2013 Conferences, Springer Berlin Heidelberg, pp. 527–541.

[10] P. Shvaiko and J. Euzenat, "Ontology matching: state of the art and future challenges", IEEE Transactions on Knowledge and Data Engineering, vol. 25(1), 2013, pp. 158–176.

[11] OMG, "CORBA Middleware Specifications", Version 3.3, Object Management Group Std., November 2012, http://www.omg.org/spec/#MW, [Online], retrieved: 02.2016.

[12] G. Söldner "Semantische Adaption von Komponenten", (semantic adaption of components), Dissertation, Friedrich-Alexander-Universität Erlangen-Nürberg, 2012.

[13] OMG, OMG Unified Modeling Language (OMG UML) Superstructure, Version 2.4.1, Object Management Group Std., August 2011, http://www.omg.org/spec/UML/2.4.1, [Online], retrieved: 06.2015.

[14] P. Jain, P. Z. Yeh, K. Verma, R. G. Vasquez, M. Damova, P. Hitzler, and A. P. Sheth, "Contextual ontology alignment of lod with an upper ontology: A case study with proton", in The Semantic Web: Research and Applications, Springer Berlin Heidelberg, 2011, pp. 80–92.

[15] D. Niebuhr, "Dependable Dynamic Adaptive Systems: Approach, Model, and Infrastructure", Clausthal-Zellerfeld, Technische Universität Clausthal, Department of Informatics, Dissertation, 2010.

[16] J. Camara, J. Martin, G. Saaün, C. Canal, and E. Pimentel, "Semi-Automatic Specification of Behavioural Service Adaptation Contracts", in Proceedings of the 7th International Workshop on Formal Engineering Approaches to Software Components and Architectures, 2010, pp. 19–34.

[17] D. Niebuhr and A. Rausch, "Guaranteeing Correctness of Component Bindings in Dynamic Adaptive Systems based on Run-time Testing", in Proceedings of the 4th Workshop on Services Integration in Pervasive Environments (SIPE 09) at the International Conference on Pervasive Services 2009, (ICSP 2009), 2009, pp. 7–12.

[18] A. Bertolino, P. Inverardi, P. Pelliccione, and M. Tivoli, "Automatic Synthesis of Behavior Protocols for Composable Web-Services", Proceedings of the the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering, 2009, pp. 141–150.

[19] J. Camara, C. Canal, J. Cubo, and J. Murillo, "An Aspect-Oriented Adaptation Framework for Dynamic Component Evolution", Electronic Notes in Theoretical Computer Science, vol. 189, 2007, pp. 21–34.

[20] H. R. Nezhad, B. Benatallah, A. Martens, F. Curbera, and F. Casti, "Semi-Automated Adaptation of Service Interactions", Proceedings of the 16th international Conference on World Wide Web, 2007, pp. 993–1002.

[21] A. Kalyanpur, D. Jimenez, S. Battle, and J. Padget, "Automatic Mapping of OWL Ontologies into Java", in F. Maurer and G. Ruhe, Proceedings of the 17th International Conference on Software Engineering and Knowledge Engineering, SEKE'2004, 2004, pp. 98–103.

[22] D. M. Yellin and R. E. Strom, "Protocol Specifications and Component Adaptors", ACM Transactions on Programming Languages and Systems, vol. 19, 1997, pp. 292–333.

 ISBN: 978-1-61208-463-3

Hints to Address Concurrency in Self-Managed Systems at the Architectural Level

A Case Study

Francesco Mazzei and Claudia Raibulet
Università degli Studi di Milano-Bicocca
DISCo - Dipartimento di Informatica, Sistemistica e Comunicazione
Milan, Italy
e-mail: f.mazzei3@campus.unimib.it, raibulet@disco.unimib.it

***Abstract*—Self-managed systems are able to perform changes by themselves on themselves during their execution due to variations occurred internally or in their execution environment. Current solutions for self-managed systems address one change at a time. In the real world, two or more changes may raise contemporaneously. Hence, they should be addressed concurrently and not sequentially. This would improve the overall performances of a system and would avoid delays in addressing changes. In this paper, we propose architecture level mechanisms to address concurrency in self-managed systems. We investigate available concurrency solutions used in non self-managed systems and adopt and adapt them for self-managed systems. We also introduce novel concepts such as: adaptation zone, adaptation need, and adaptation level. To apply and validate our solution a case study in a Web banking context has been developed.**

Keywords-self-adaptivity; concurrency; architecture.

I. INTRODUCTION

Over the years, systems have grown in size and complexity and many of them are required to run continuously. Therefore, it has become important to develop systems that are able to manage and adapt themselves at runtime in response to changing requirements and environmental conditions. Examples of situations in which self-managed systems may offer a valid solution are identified in [12]: (1) systems should run continuously in the presence of components' faults, variability in resources, or variability of users' needs, (2) administrative overheads should be reduced allowing smooth operation with minimal human oversight, and (3) systems should provide various levels of services to different users depending on their needs and context.

Current solutions consider that systems address one change at a time. In the real world, two or more changes may occur contemporaneously. Several issues may raise here: is it possible to address them concurrently? If yes, under which conditions (considering that modifications are performed at runtime and systems should provide the functionalities for which they have been designed independent of the computation performed for their self-management issues). If no, how to establish their priority for changes' execution? Hence, how to address two or more changes at a time in a self-managing system?

Engineering self-managed systems is not a trivial task [3][5][9]. Addressing one change at a time is complex. However, it is easier to reason about one change at a time: no synchronization or consistency issues should be considered. Addressing two or more changes at a time becomes a significant issue. Each change should leave the self-managed system in a stable state. A question may raise here: is it worth the effort considering the possible benefits of addressing concurrent issues? David Garlan mentions two possible benefits: (1) improvement of the performances of the self-managed systems through the parallelization of the changes, and (2) provision of an immediate feedback when change needs raise. For example, if a self-optimization task is running in the system, and a self-healing issue occurs in the meantime, it is desirable that the last is addressed immediately independent of the fact that the first continues its execution or should be stopped or finished immediately.

As far as concerns our knowledge, there is no available solution for engineering self-managed systems that implements concurrency mechanisms to manage contemporaneously two or more changes. The closest solutions take into consideration multiple objectives when deciding which change to perform in the system [2]. However, the multiple objectives are summarized into a single change.

The objective of this paper is to investigate the available concurrency mechanisms which have been defined for traditional systems and which can be adopted and/or adapted for self-adaptive systems. Examples of mechanisms for addressing concurrency issues include prioritization, scheduling, architectural and design patterns. This work aims also to validate the identified solutions through a common and actual case study.

The rest of the paper is organized as follows. Section II presents the main concepts used in our solution. Section III presents a case study using the previously introduced concepts. The paper ends with the Conclusions and Future Developments presented in Section IV.

II. MAIN CONCEPTS OF OUR SOLUTION

Prof. David Garlan mentioned concurrency [7] as one of the future challenges of self-managed systems at SEAMS 2013. The potential benefits of exploiting concurrency concern performance and rapid response when new self-managing issues arise. He also suggested three ideas on how to manage concurrency in such systems:

 ISBN: 978-1-61208-463-3

- non-interference guarantee between concurrent adaptations;
- possible interruptions of ongoing adaptations, when higher-priority adaptations occur;
- possible finish of unproductive adaptations.

Starting from these course-grained ideas, we try to expand them and to provide fine-course possible hints.

In the remaining of this section, we introduce the main concepts on which our solution is based: Monitor-Analyze-Plan-Execute (MAPE) loop, adaptation zone, adaptation process, adaptation need, and adaptation level.

A. The MAPE Loop

Architectural-based solutions for self-managed systems exploit feedback concepts and control mechanisms [3][5]. The feedback enables a system to understand its current state and its execution environment by monitoring itself and its surrounding world. To achieve this the system collects meaningful data through various sensors and/or mechanisms, data which is further analyzed by the system. The control enables a system to be active and perform changes on itself in correspondence of variations in its execution environment. To achieve this a system chooses the most appropriate changes to be performed in its current state and implements mechanisms to apply the identified changes.

The feedback control loops consist in the following four steps: Monitoring, Analyzing, Planning, and Executing (MAPE) [5].

B. Adaptation Zone

An *adaptation zone* indicates a part of a system which may be subject to changes at runtime. In other words, an adaptation zone indicates the co-related elements of a system which may be involved in a type of adaptation at runtime. A zone is a mutable, a dynamic part of a system.

At the architectural level, an *architectural adaptation zone* may be composed of components, packages, classes, interfaces. Each architectural adaptation zone is identified using a name that reminds the type of adaptation in that zone. It is possible to use, for example, the name of each adaptation use case for the corresponding zones.

From the concurrency point of view, if two or more adaptations occur in disjoint architectural adaptation zones they may be executed in parallel without any further concurrency issue. Otherwise, if two or more adaptations occur in the same architectural adaptation zone, concurrent issues, such as shared resources problem, should be considered and a mutual exclusion solution is needed. To address this issue we define the *runtime adaptation zone*. At runtime, many instances of a system's element may be created (e.g., objects). An adaptation may use only part of the instances available in an architectural adaptation zone. Hence, we introduce the concept of runtime adaptation zone to group together the instances of the system's elements of an adaptation zone used actually in an adaptation.

A runtime adaptation zone may be in one of the following states:

- green, meaning that in the runtime adaptation zone every object is unlocked, or rather no adaptations are running in the zone; when an adaptation finds the green color, it will run and use objects without any constraint;
- yellow, meaning that in the runtime adaptation zone one or more objects are in use by other adaptation(s); an adaptation may run in this zone only if it uses objects that are not locked by the other adaptation(s);
- red, meaning that in the runtime adaptation zone every object is locked, or rather one or more adaptations are running in this zone using all objects; when an adaptation finds the red color, it will wait until the objects it plans to use will be unlocked.

A runtime adaptation zone may modify its state going in one of the remaining two states without following any particular sequence.

C. Adaptation Process

An adaptation consists in a change in a system. It is the result of a feedback control process composed of four main steps: monitoring, analyzing, planning, and executing. Each of these steps may be complex and may require several entities for its implementation. In our solution, we have defined four managers, each supervising one step of the adaptation process. The Adaptation Monitor gathers the data describing the current state of the execution environment of a system (both the system itself and its external world). The Adaptation Analyzer verifies the current state of a system and identifies the variations which may require a change in a system. These two entities must run for the entire lifetime of the system and they continuously or periodically check the state of a system.

From the concurrency point of view, there are no particular issues in these two steps. The monitoring step may gather concurrently data from various sources regarding various aspects. The analyzing step may verify concurrently various data to reveal variations.

The planning step is managed by the Adaptation Planner. The planning step identifies the change to be performed and the strategy to apply the identified change in a system. A strategy is composed of a set of operations required to adapt a system to a need starting from its current state. The Adaptation Planner is the manager of the adaptation strategies. To develop interchangeable strategies, the Strategy design pattern is used.

From the concurrency point of view, the Adaptation Planner may manage in parallel two or more adaptations each considered separately. It works similarly to the Adaptation Monitor and Adaptation Analyzer with the difference that it is activated only when an adaptation is needed in a system.

When the Adaptation Planner has decided which is the most appropriate strategy to be applied for the current adaptation, it has to be performed. This is the most

 ISBN: 978-1-61208-463-3

important part of the concurrent adaptation mechanism considered in this paper. In self-managed systems in which there is no concurrency, the adaptation is performed without any particular issues and complexity. But, in this case, more than one adaptation may be needed at the same time. Therefore, this part of the solution must have specific attributes that characterize every adaptation in order to compare them. Hence, we have introduced an entity that represents the execution of an adaptation: the Adaptation Performer.

The Adaptation Performer is the entity that has the responsibility to apply the strategy selected by the Adaptation Planner. It is the element that applies in a system those operations that compose the strategy (see Figure 1).

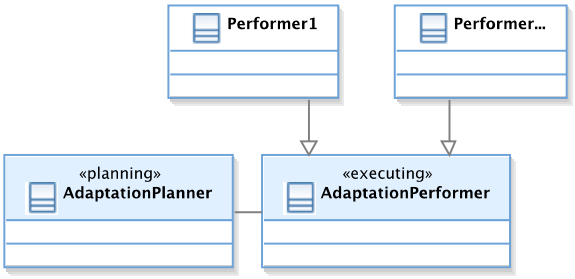


Figure 1. The Adaptation Performer and the Adaptation Planner

From the concurrency point of view, it has been asserted that each adaptation involves only a specific architectural zone, hence one of the Adaptation Performer's attribute is the architectural adaptation zone in which the adaptation has to be applied (see Figure 2). Two or more Adaptation Performers can adjust a system concurrently if they involve different architectural adaptation zones. Furthermore, due to the runtime adaptation zones, if performers involve the same architectural zone, but not the same set of objects, they can run concurrently. However, when they need the same zones and the same objects, they must be compared among them, to determine which should be run, interrupted, or stopped. An entity like a scheduler can do the comparison. To achieve this goal, it is necessary to understand the characteristics of each adaptation and which of those it is useful for the comparison. It has been defined a particular attribute that groups some important characteristics, and it is called Adaptation Need (see Section II.D).

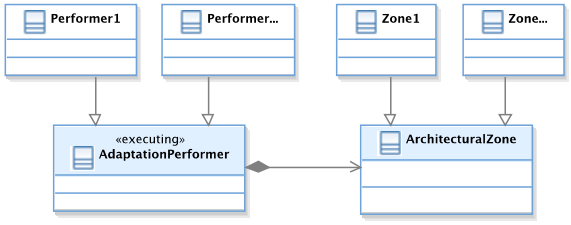


Figure 2. Adaptation Performer with the Architectural Adaptation Zones

D. *Adaptation Need*

An adaptation may be of various types: self-configuration, self-optimization, self-protection, self-healing. and so on. Each of these types has a different importance for a system. Generally, a self-healing or a self-protection adaptation have the greater importance. Obviously, this importance depends strongly on the application domain. Based on these affirmations, the first attribute that characterizes an adaptation is its *type*. Based on the *type* attribute it is possible to define a hierarchy of priorities for adaptations for each system.

With the *type* attribute a first comparison is done among Adaptation Performers. However, if two or more performers have the same type, other attributes are needed to prioritize them. A second attribute consists in the adaptation *Strategy* which has been chosen by the Adaptation Planner. A strategy has associated a static priority (e.g., as the priority of the create, update, insert, delete operations in a database). Further, a strategy spends an estimated time to perform its operations, so it has a considerable importance for the comparison. Thus, *Time* is the third and last element that characterizes an adaptation need. It estimates how much time a strategy needs to be completely performed (see Figure 3). Based on the application domain, the highest priority may be assigned to the strategy having the minor estimated time, or the major estimated time.

To summarize, two steps are performed to compare two or more Adaptation Performers:

- step 1: use type to compare two or more Adaptation Performers;
- step 2: if type is identical, use strategy and time to compare two or more Adaptation Performers.

Due to this two steps comparison, it is possible to decide which process has to be run, interrupted, or stopped.

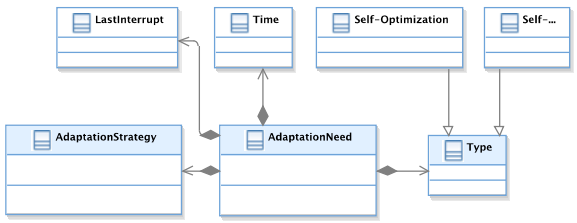


Figure 3. The Adaptation Need with Type, Strategy and Time

If a performer has to be interrupted, a mechanism to interrupt and resume it, is needed. A scheduler can do the comparison between the performers that will run in the same architectural adaptation zone with the same objects and it can choose if a performer has to be interrupted and then resume it. It is possible that the scheduler interrupts more than one Adaptation Performers, so it must have a queue of interrupted processes. To allow the scheduler to choose which performer has to be resumed, the Adaptation Need requires another attribute, called *last interrupt*.

 ISBN: 978-1-61208-463-3

Last interrupt (see Figure 3) defines the last time that a performer has been interrupted (e.g., the timestamp). Every time the scheduler has to choose which Adaptation Performer has to be resumed, it sorts the queue of performers by *last interrupt* and chooses the performer that has the oldest timestamp.

However, in an application there could be more aspects to take into account to resume a performer. Therefore, for each application the last interrupt can be used in combination with other application's elements.

E. Adaptation Level

If one or more processes have to be interrupted or stopped, a stability problem may occur in the system. An adaptation cannot be interrupted or stopped anytime, but it must do this leaving a system in a stable state.

The Adaptation Performer adapts the system performing the Adaptation Strategy's operations. To guarantee the stability, any operation can become an atomic step, in order to allow the interruption (or the stop) after the ending of an operation and before the beginning of another. Clearly, if some sub-steps compose an operation, they must be all executed without stopping. In Stitch [1] language, strategies are composed by tactics. For our solution, we consider that those atomic operations, which compose an Adaptation Strategy, are called tactics.

Furthermore, it is also possible to estimate how much a system has been adapted, according to the number of tactics performed. This is an adaptation value, and it was named *Adaptation Level*.

For example, if three tactics compose the Adaptation Performer strategies, the Adaptation Level can be:

- Low-level: the performer has completed only one tactic;
- Medium-level: the performer has completed two tactics;
- High-Level: the performer has completed three tactics.

This is an easy example, but the Adaptation Level can be usually divided into the three levels. To estimate at which level a performer has adapted a system, it is possible to calculate with percentage notation the operations performed (e.g., 34%, 67%, 100%) or any other solution that the system's engineer considers appropriate. The Adaptation Level is useful to provide information about a system and a system's adaptation. Furthermore, a system may take information about its adaptation, to dynamically add, modify or remove an Adaptation Strategy and modify the estimated Time of an adaptation.

The objects in each zone are in a stable state if there are no Adaptation Performers running tactics over them. Hence, there are three definitions of stable states, object for the objects, local for the zone, and global for the entire system.

- Object stable state: an object is in a stable state if it is not used by an Adaptation Performer;
- Local stable state: a zone is in a stable state if no object is used by an Adaptation Performer (the runtime zone is in a green state);
- Global stable state: the system is in a stable state if all the runtime zones are in a green state.

Therefore, an Adaptation Performer can be interrupted between a tactic and another, when it leaves all the involved objects in a stable state. It is important to understand that each performer has not the responsibility of the entire system's stability (or the zones' stability), but it only has to care about the objects that it involves.

So with all of these definitions, an Adaptation Performer can perform the strategy's operations and it can be interrupted or stopped when necessary.

III. THE UNIBANK CASE STUDY

The solution presented in the previous section has been validated through a case study, a home banking system, called Unibank. The solution implements two possible adaptation types: an architectural one (e.g., addition/removal of servers) and a content one (e.g., visualization of textual and/or multimedia content). The current version of the solution enables (1) the concurrent execution of two or more adaptation processes if they involve different architectural zones, (2) the interruption of an adaptation process if another one arise in the same adaptation zone with a higher priority, or (3) the ending of an adaptation process under certain conditions.

Unibank is a home banking system that provides several services to its customers (e.g., register, create a bank account, visualize the history of the operations performed, do a transfer, require a credit card). In such a system, there are various aspects which can be properly addressed through self-managed solutions. For example, the variations in the system may concern the number of clients' requests, the number of replicated servers the system uses, the available bandwidth, the type of the clients' devices used to communicate with the system.

In the remaining of this section, we present two types of adaptations in which concurrency based concepts are efficiently exploited. Both types of adaptations are triggered by the variations of the number of clients' request. To ensure a constant level of quality, the system may decide to add/remove a replicated server, and/or, vary the quality of the content visualization (e.g., visualize multimedia information and/or only textual information). Further, we introduce briefly the architectural aspects of our solution.

A. Variation of the Number of Servers

The number of clients' requests continuously change, from a low level to a high level. It means that if the number of requests is high, the system needs a greater number of replicated servers to handle them. Vice versa, if the number of requests is low, the system needs a lower number of servers, to reduce the services' costs. It is also possible that one or more servers crash, so other servers are needed to handle the current number of requests.

Every home banking system must guarantee a 365/24 service, regardless of the requests' level and servers status. However, every system has a cost and a budget limit, so

 ISBN: 978-1-61208-463-3

they have also a number of running servers limit. In fact, there is a limited server pool from which to add or remove a specific server.

Hence, in an architectural server-based adaptation, there are two strategies: add one or more servers, and remove one or more servers. This type of adaptation and these two strategies are very similar in their execution. In the following, one of them is described step-by-step as an example:

1) Adding servers according to clients' requests

1. The Adaptation Monitor supervises the number of the clients' requests.
2. The Adaptation Analyzer reveals an increase of the number of clients' requests.
3. The Adaptation Planner chooses a strategy: add a specific number of servers.
4. In the execution step, the Adaptation Performer adds every server needed (if the pool has available the requested number of servers).

B. Variation of the Quality of Content Visualization

For many reasons, the architectural server-based adaptation is not always available (e.g., all the available servers are functioning). Therefore, if the number of clients' requests increases and the adaptation process cannot add replicated servers, it may vary the quality of the content visualization to allow the handling of every request. Vice versa, if the number of requests decreases the adaptation process can improve the quality of the content visualization.

Hence, in a variation of quality of content visualization adaptation there are two strategies: improve the quality of the content visualization, and reduce the quality of the content visualization.

This type of adaptation and these two strategies are very similar in their execution. In the following, one of them is described step-by-step as an example:

1) Improve the quality of contents visualization according to the number of clients' requests

1. The number of client's requests has decreased;
2. When the Adaptation Process starts in the monitoring step, it reveals the number of requests;
3. The process passes to the analyzing step and it now knows that the actual number of servers and quality of contents visualization are too much to handle requests;
4. In the planning step, the Adaptation Process chooses a strategy: improve the quality of contents visualization;
5. In the execution step, the process puts the quality of contents visualization in a higher level.

C. Architectural Aspects

The architecture of the Unibank system is presented in Figure 4. The two grey elements indicate two different architectural adaptation zones. Currently, these architectural adaptation zones are defined statically at design time.

The Adaptation Monitor and the Adaptation Analyzer have been implemented as Singletons because they are quite simple in this case study: they monitor and analyze the number of the clients' requests.

Once that the data is checked and an adaptation is required, the Adaptation Process goes to the planning step. The Adaptation Planner is the element that represents the planning step, thus it decides how to adapt the system, according to the result of the Adaptation Analyzer.

To allow the handling of multiple adaptations, the Adaptation Process creates an independent (asynchronous) instance of the planner. Hence, after the analyzing step, the Adaptation Planner continues the adaptation process. The Adaptation Planner is a thread that is created only if an adaptation is needed.

When the Adaptation Planner is created, it has to choose how the adaptation has to be performed. It has been initially decided that there are only two types of adaptation in this case study (architectural server-based adaptation and quality of the content visualization) and each type has only two strategies to be performed (add/remove server and improve/reduce quality). Therefore, the Adaptation Planner has a link to those strategies, and it chooses (1) the kind of adaptation required, and (2) the suitable strategies.

To choose the strategy, the Adaptation Planner uses the result that the Adaptation Analyzer has returned after the analyzing step. Even if, in this case study, there are only four strategies, it was designed a reflection mechanism to allow a dynamic update of strategies. Once the strategy has been selected, the planner creates an instance of it. The Adaptation Strategy prepares the object that will perform the adaptation, the Adaptation Performer.

Such as for the strategies, there is not only one Adaptation Performer. Each performer has to run in a particular zone of the system and it uses only that zone's objects. For this reason, there was designed a Strategy design pattern to implement the performers. Each performer is associated to one zone, so in this case study we have two Adaptation Performers: ServerAdaptationPerformer and JspAdaptationPerformer.

The Adaptation Performer executes the strategy tactic-by-tactic [1][6], so that it can be stopped between a tactic and another. Tactics are atomic operations that compose a strategy and the number of tactics performed by a performer gives the Adaptation Level. Therefore, appropriate tactics were defined for the Adaptation Strategies. For the quality of content visualization strategies (improve quality and reduce quality), a tactic was implemented for each reachable visualization quality level, which improves or reduces the quality of the contents. This case study was designed with three levels of quality of content visualization. Hence, the Adaptation Level is incremented for each improved or reduced quality level gained. For the architectural server-based structural strategies (add server and remove server) it was defined a tactic based on the number of servers to add or remove. If a performer has to add three servers, the tactics that compose this strategy are three. Hence, the Adaptation Level is incremented for each added or removed server.

To handle the concurrency adaptations and so multiple performers, which will run concurrently to adapt the system,

 ISBN: 978-1-61208-463-3

in Unibank it was designed an Adaptation Scheduler. The scheduler has the objective of handle every Adaptation Process; it compares them to choose which adaptation process has to run, to stop or to interrupt. Two processes that involve two different architectural zones can run concurrently, each scheduler handles only the processes that run in a specific zone.

Each scheduler is a Singleton. The Adaptation Planner receives the Adaptation Performer from the Adaptation Strategy and then it communicates with the appropriate scheduler to add the performer. The Adaptation Scheduler uses the performer's Adaptation Need to compare it with the other performers that are running in a specific area.

The routine used to compare and manage the performers is based on the runtime adaptation zones and Adaptation Need concept. To simplify, only the performers' stops were considered and not the interruptions. However, it is only one of the multiple solutions that are achievable with these notions.

When the Adaptation Planner adds a performer to the scheduler, the scheduler starts to check the performer's attributes. The Adaptation Scheduler checks first how many performers are running in its zone. If no performer is running (so the runtime zone is in a green state) the scheduler can run the new Adaptation Performer, otherwise it has to check which objects the new performer has to use to adapt the system (the runtime is in a yellow or red state). If the objects, which the new performer has to involve, are in use by other performers, the Adaptation Scheduler has to compare the priority of the new performer with those of the other performers.

If the priority of the other running performers is lower, then the scheduler has to interrupt or stop them and to start the execution of the new performer. Recall that an Adaptation Performer cannot be interrupted or stopped anytime, but it can be stopped after the end of a tactic and the start of another, that is because every performer must leave the objects in a stable state.

If the priority of the other running performers is higher, then the scheduler has to reject the adaptation, because other most important performers (and so adaptations) are running.

If the priority is the same, the scheduler has to compare the Time and the Strategy to choose which performer has the highest priority. This application has a strategy's priority hierarchy for each type of adaptation. For server structural adaptation: add server and remove server. And for quality of contents visualization adaptation: improve quality of contents and reduce quality of contents.

Therefore, when the type of two performers is the same, the Adaptation Scheduler compares the strategy and then uses the same previous routine. When also the strategies are the same, the scheduler gives priority to the shorter performer according to its Time attribute. When two performers are completely equal, the Adaptation Scheduler chooses the performer that was added first.

IV. CONCLUSIONS AND FURTHER WORK

The work presented in this paper has addressed concurrency issues in self-adaptive systems by focusing on how two or more adaptation needs which occur in the same time interval can be properly managed. The issues raised by this topic are mainly due to the fact that the system should address two or more adaptations by itself during its execution. Each adaptation should leave the system in a stable state. Making two changes in a system may be risky also when the system is in the development phase, while during its execution is a challenge.

Several concepts have been used in this paper such as adaptation need, adaptation process, adaptation level, adaptation strategies and tactics, and priorities. Further novel concepts have been introduced for addressing concurrency: architectural adaptation zone and runtime adaptation zone. Our solution will be further refined by considering issues and solutions proposed in various fields such as Self-Organizing Networks (SON) mechanisms for future wireless networks, i.e., LTE [4].

The solution has been validated through a case study called Unibank, implementing a home banking application. Two types of adaptations have been considered: architectural (e.g., the changing number of the used servers) and content-based (e.g., the changing of the content type - textual and multimedia - displayed to the user). The solution enables (1) the concurrent execution of two or more adaptation processes if they involve different architectural zones, (2) the interruption of an adaptation process if another one arise in the same runtime adaptation zone with a higher priority, or (3) the ending of an adaptation process under certain conditions. In the further work, we plan to validate our solution in other case studies and application domains and to use available tools and approaches for formal validation.

In this paper, we have defined architectural zones, architectural levels, adaptation strategies, and adaptation types statically at design time. A future work plans to introduce flexibility in the definition of architectural zones, and enable their definition and/or modifications at runtime.

Another further work concerns the availability of various access devices. Today every person has more types of Internet-connected devices ranging from smartphones and tablets to laptops and desktops. To overcome the visualization problems for different types of devices, we use Bootstrap, which supports responsive Web design. The layout of Web pages adjust dynamically, taking into account the characteristics of the device used for the access of Unibank. This kind of adaptation is different from the ones presented in this paper, being not jet included in the adaptation process of Unibank.

Finally, we plan to measure the performances of our solution addressing concurrent issues considering quality attributes [8][10] and software metrics [11].

 ISBN: 978-1-61208-463-3

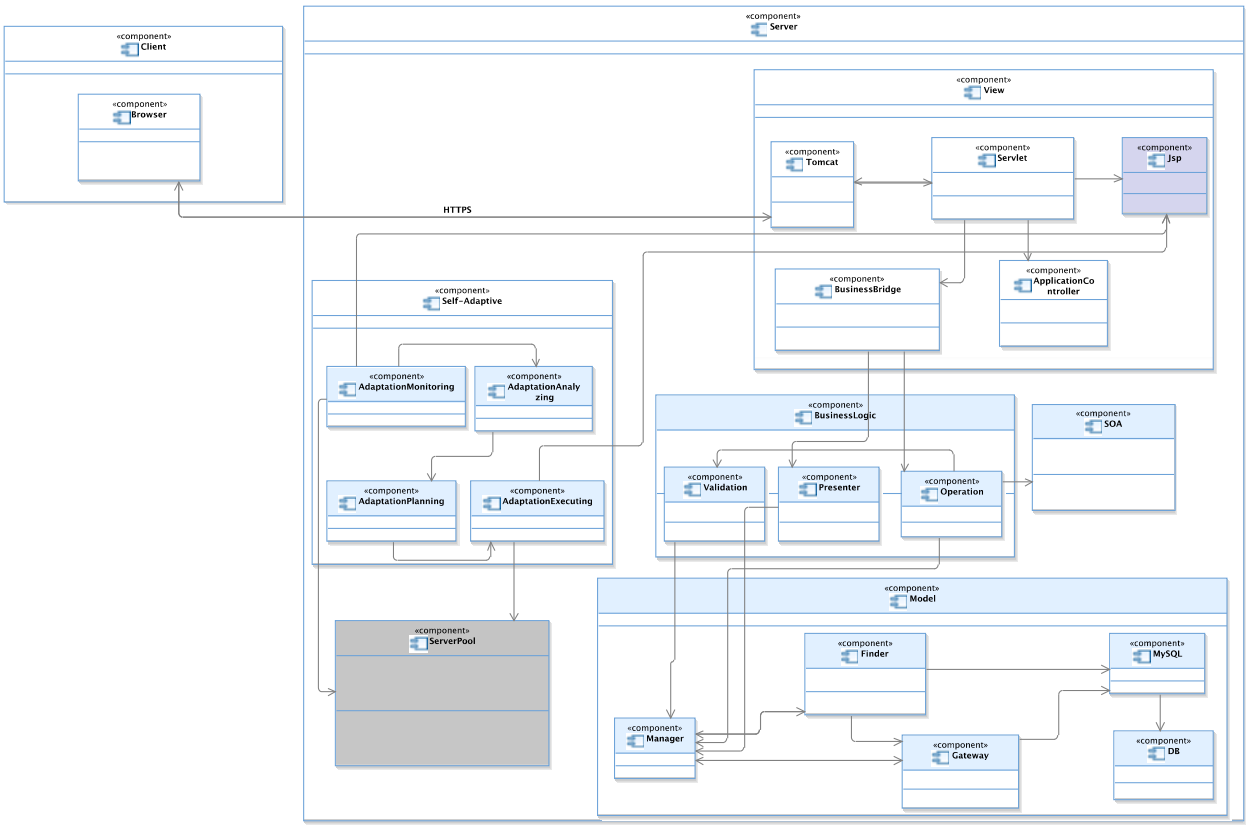


Figure 4. Unibank Architecture

REFERENCES

[1] S. W. Cheng and D. Garlan, "Stitch: A language for architecture-based self-adaptation". Journal of Systems and Software, vol. 85, 2012, pp. 1860-2875.

[2] S. W. Cheng, D. Garlan, and B. Schmerl, "Architecture-based Self-Adaptiation in the Presence of Multiple Objectives", SEAMS 2006, pp. 2-8.

[3] B. H. C. Cheng, R. de Lemos, H. Giese, P. Inverardi, and J. Magee, Software Engineering for Self-Adaptive Systems. LNCS 5525, Springer, 2009.

[4] L. Ciavaglia, Z. Altman, E. Patouni, A. Kaloxylos, N. Alonistioti, K. Tsagkaris, P. Vlacheas, and P. Demestichas, "Coordination of Self-Organizing Network Mechanisms: Framework and Enablers", Mobile Networks and Management, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Vol 97, 2012, pp 174-184.

[5] R. de Lemos, H. Giese, H. Muller, and M. Shaw, Software Engineering for Self-Adaptive Systems II, Lecture Notes in Computer Science 7475, Spinger, 2012.

[6] D. Garlan, S. W. Cheng, A. C. Huang, and B. Schmerl, P. Steenkiste, "Rainbow: Architecture-based Self-Adaptation with Reusable Infrastructure". IEEE Computer, Vol. 37, No. 10, IEEE Computer Society Press, 2004, pp. 46-54.

[7] K. E. Harper, J. Zheng, and S. Mahate, "Experiences in Initiating Concurrency Software Research Efforts". 32nd International Conference on Software Engineering, vol. 2, 2010, pp. 139-148.

[8] S. Neti, and H. Müller. "Quality Criteria and Analysis Framework for Self-Healing Systems", ICSE Workshop on Software Engineering for Adaptive and Self-Management Systems, 2007.

[9] C. Raibulet, "Facets of Adaptivity". 2nd European Conference on Software Architecture, LNCS 5292, 2008, pp. 342-345.

[10] C. Raibulet. "Hints on Quality Evaluation of Self-* Systems", 8th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, London, UK, September 8th-12th, 2014.

[11] P. Reinecke, K. Wolter and A. Van Moorsel. Evaluating the Adaptivity of Computing Systems, Performance Evaluation Journal, Vol. 67, pp. 676-693. 2010.

[12] T. Seceleanu and D. Garlan, "Synchronized Architectures for Adaptive Systems". 29th Annual International Computer Software and Applications Conference. Edinburgh, UK, 2005, pp. 146-151.

 ISBN: 978-1-61208-463-3