ICAS 2011

The Seventh International Conference on Autonomic and Autonomous Systems

May 22-27, 2011

Venice/Mestre, Italy

ICAS 2011 Editors

Alex Galis, University College London, UK

Bruno Dillenseger, Orange Labs, France

ICAS 2011

Foreword

The Seventh International Conference on Autonomic and Autonomous Systems (ICAS 2011), held between May 22 -27, 2011 in Venice, Italy, was a multi-track event covering related topics on theory and practice on systems automation, autonomous systems and autonomic computing.

The main tracks referred to the general concepts of systems automation, and methodologies and techniques for designing, implementing and deploying autonomous systems. Next tracks developed around design and deployment of context-aware networks, services and applications, and the design and management of self-behavioral networks and services. Also considered were monitoring, control, and management of autonomous self-aware and context-aware systems and topics dedicated to specific autonomous entities, namely, satellite systems, nomadic code systems, mobile networks, and robots. It has been recognized that modeling (in all forms this activity is known) is the fundamental for autonomous subsystems, as both managed and management entities must communicate and understand each other. Small-scale and large-scale virtualization and model-driven architecture, as well as management challenges in such architectures were considered. Autonomic features and autonomy requires a fundamental theory behind and solid control mechanisms. These topics give credit to specific advanced practical and theoretical aspects that allow subsystem to expose complex behavior. It was aimed to expose specific advancements on theory and tool in supporting advanced autonomous systems. Domain case studies (policy, mobility, survivability, privacy, etc.) and specific technology (wireless, wireline, optical, e-commerce, banking, etc.) case studies were targeted. A special track on mobile environments was indented to cover examples and aspects from mobile systems, networks, codes, and robotics.

Pervasive services and mobile computing are emerging as the next computing paradigm in which infrastructure and services are seamlessly available anywhere, anytime, and in any format. This move to a mobile and pervasive environment raises new opportunities and demands on the underlying systems. In particular, they need to be adaptive, self-adaptive, and context-aware.

Adaptive and self-management context-aware systems are difficult to create, they must be able to understand context information and dynamically change their behavior at runtime according to the context. Context information can include the user location, his preferences, his activities, the environmental conditions and the availability of computing and communication resources. Dynamic reconfiguration of the context aware systems can generate inconsistencies as well as integrity problems, and combinatorial explosion of possible variants of these systems with a high degree of variability can introduce great complexity.

Traditionally, user interface design is a knowledge-intensive task complying with specific domains, yet being user friendly. Besides operational requirements, design recommendations refer to standards of the application domain or corporate guidelines.

Commonly there is a set of general user interface guidelines; the challenge is due to a need for cross-team expertise. Required knowledge differs from one application domain to another, and the core knowledge is subject to constant changes and to individual perception and skills.

Passive approaches allow designers to initiate the search for information in a knowledge-database to make accessible the design information for designers during the design process. Active approaches, e.g., constraints and critics, have been also developed and tested. These mechanisms deliver information (critics) or restrict the design space (constraints) actively, according to the rules and guidelines. Active and passive approaches are usually combined to capture a useful user interface design.

All these points posed considerable technical challenges and make self-adaptable context-aware systems costly to implement. These technical challenges led the context-aware system developers to use improved and new concepts for specifying and modeling these systems to ensure quality and to reduce the development effort and costs.

SYSAT Advances in system automation
AUTSY Theory and Practice of Autonomous Systems
AWARE Design and Deployment of Context-awareness Networks, Services and Applications
AUTONOMIC Autonomic Computing: Design and Management of Self-behavioral Networks and Services
CLOUD Cloud computing and Virtualization
MCMAC Monitoring, Control, and Management of Autonomous Self-aware
CASES Automation in specialized mobile environments
ALCOC Algorithms and theory for control and computation
MODEL Modeling, virtualization, any-on-demand, MDA, SOA
SELF Self-adaptability and self-management of context-aware systems
KUI Knowledge-based user interface
AMMO Adaptive management and mobility

We welcomed technical papers presenting research and practical results, position papers addressing the pros and cons of specific proposals, such as those being discussed in the standard forums or in industry consortia, survey papers addressing the key problems and solutions on any of the above topics short papers on work in progress, and panel proposals.

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

We hope that ICAS 2011 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in autonomic and autonomous systems.

We are certain that the participants found the event useful and communications very open. We also hope the attendees enjoyed the beautiful surroundings of Venice.

ICAS 2011 Chairs
Michael Bauer, The University of Western Ontario - London, Canada
Radu Calinescu, Aston University, UK
Larbi Esmahi, Athabasca University, Canada
Alex Galis, University College London, UK

ICAS 2011

Committee

ICAS Advisory Chairs

Michael Bauer, The University of Western Ontario - London, Canada
Radu Calinescu, Aston University, UK
Larbi Esmahi, Athabasca University, Canada
Alex Galis, University College London, UK
Michael Grottke, University of Erlangen-Nuremberg, Germany
Antonio Liotta, Eindhoven University of Technology, The Netherlands

ICAS Industry/Research Chairs

Andrew J. Cowell, Pacific Northwest National Laboratory, USA
Bruno Dillenseger, Orange Labs, France
Kazuo Iwano, IBM Japan, Japan
Marius Slavescu, Elegant Computing Services Inc., Canada
Martin Zach, Communications, Media and Technology, Siemens AG, Austria

ICAS 2011 Technical Program Committee

Jemal H. Abawajy, Deakin University, Australia
Noura Achour, USTHB - Algiers, Algeria
Matthew Adigun, University of Zululand - Kwadlangezwa, South Africa
Jérémie Albert, University of Bordeaux, France
Tomlinson Allan, University of London, UK
Javier Alonso, Technical University of Catalonia, Spain
Razvan Andonie, Central Washington University - Ellensburg, USA
Françoise André, University of Rennes 1, France
Richard Anthony, University of Greenwich, UK
Eva Ibarrola Armendariz, Escuela Técnica Superior de Ingeniería de Bilbao, Spain
Michael Bauer, The University of Western Ontario -London, Canada
Julita Bermejo-Alonso, Universidad Politecnica de Madrid, Spain
Philippe Besnard, IRIT/Université Paul Sabatier - Toulouse, France
Ateet Bhalla, NRI Institute of Information Science and Technology - Bhopal, India
Karsten Böhm, Fachhochschule Kufstein, Austria
David W Bustard, University of Ulster, UK
Radu Calinescu, Aston University, UK
Paolo Campegiani, University of Roma Tor Vergata, Italy
Marcelino Campos Oliveira Silva, Federal University of Rio de Janeiro, Brazil
Sara Casolari, Università di Modena e Reggio Emilia, Italy
Michael Cebulla, KRALLMANN AG, Germany
Fernando Cerdan, Universidad Politecnica de Cartagena, Spain
Radovan Cervenka, Whitestein Technologies, Slovak Republic
Carlos Cetina, Technical University of Valencia, Spain

Table of Contents

Human/Robot Multi-initiative Setups for Assembly Cells 1
Dan Gadensgaard and David Alan Bourne

A Distributed E-health Model Using Mobile Agents 7
Ali Akbar Pouyan, Sadegh Ekrami, and Momeneh Taban

Modular Structure of Neural Networks for Classification of Wooden Surfaces with PLC Industrial Implementation 13
Irina Topalova and Alexander Tzokev

Efficient Web Based Monitoring and Control System 18
Ahmed Mohamed and Hosny Abbas

A Distributed Multi-Agent Control Model for Railway Transportation System 24
Ali Akbar Pouyan, Sadegh Ekrami, and Momeneh Taban

Flexible Adaptation Loop for Component-based SOA Applications 29
Cristian Ruz, Francoise Baude, and Bastien Sauvan

Introduction of Self Optimization Features in a Selfbenchmarking Architecture 37
El Hachemi Bendahmane, Bruno Dillenseger, and Patrice Moreaux

An Introduction to Evolving Systems: Adaptive Key Component Control 43
Mark Balas and Susan Frost

Multi-agent Hierarchical System Based on ElGamal Decryption Algorithm With K+1 Access Levels 51
Iulia Stefan, Laura Vegh, George Mois, Stelian Flonta, Szilard Enyedi, and Liviu Miclea

Using a Network of Untrusted Computers for Secure Computing 57
Michal Maly

Reduced Order Modeling of Linear MIMO Systems Using Particle Swarm Optimization 62
Dia Abu-Al-Nadi, Othman Alsmadi, and Zaer Abo-Hammour

Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow 67
Xavier Dutreilh, Sergey Kirgizov, Olga Melekhova, Jacques Malenfant, Nicolas Rivierre, and Isis Truck

Improving the Development Process for Teleo-Reactive Programming Through Advanced Composition 75
James Hawthorne, Richard Anthony, and Miltos Petridis

VCE - A Versatile Cloud Environment for Scientific Applications 81
Martin Koehler and Siegfried Benkner

Self-adaptive tuning of dynamic changing problem solving: a first step to endogenous control in Multi-Agents Based Problem Solvers 88
Gael Clair, Frederic Armetta, and Salima Hassas

Topology Control of Mobile Robot Networks for Optimal Coverage and Connectivity using Irregular Hexagonalization 94
Karthik Mahesh Varadarajan and Markus Vincze

UAV Speed Estimation With Multi-bit Quantizer in Adaptive Power Control 100
Hyeon-Cheol Lee

A Simple Language for Describing Autonomous Agent Behaviors 105
Philippe Codognet

Mobile Robots Path Planning using Genetic Algorithms 111
Nouara Achour and Mohamed Chaalal

Utility Function-based Workload Management for DBMSs 116
Mingyi Zhang, Baoning Niu, Patrick Martin, Wendy Powley, Paul Bird, and Keith McDonald

A Framework for Certifying Autonomic Computing Systems 122
Haffiz Shuaib, Richard Anthony, and Mariusz Pelc

The Challenge of Validation for Autonomic and Self-Managing Systems 128
Thaddeus Eze, Richard J. Anthony, Chris Walshaw, and Alan Soper

Nature-inspired Self-organization in P2P Ad hoc Grids 134
Tariq Abdullah and Koen Bertels

Aspects of Innateness and Introspection in Artificial Agents 140
Chris White, David Bell, and Weiru Liu

A Self-diagnosis Algorithm Based on Causal Graphs 146
Jingxian Lu, Christophe Dousson, and Francine Krief

Convergence singularities: As man becomes machine and machine becomes man 152
Carl Adams

An Autonomic Security Mechanism Based on Novelty Detection and Concept Drift 158
Gesiel Rios Lopes, Andre L. V. Coelho, and Raimir Holanda Filho

Dynamic Model-based Management of a Service-Oriented Infrastructure 164
Felix Cuadrado, Rodrigo Garcia-Carmona, and Juan C. Duenas

A Dynamic Service Composition using Semantic Ratiocination on Integrated Home Network System 170
Junsoo Kim, Junya Nakata, Takashi Okada, Marios Sioutis, Azman Osman Lim, and Yasuo Tan

A Methodology for Household Appliances Behaviour Recognition in AmI Systems 175
Ovidiu Aritoni and Viorel Negru

Collaboration Language for Social Information Engineering 179
Kurt Englmeier, Josiane Mothe, Fionn Murtagh, Javier Pereira, and Duska Rosenberg

Human/Robot Multi-initiative Setups for Assembly Cells

Dan Gadensgaard
Department of Mechanical and Manufacturing Engineering
Aalborg University
Aalborg, Denmark
Email: dan.gadensgaard@gmail.com

David Alan Bourne
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA, USA
Email: db@ri.cmu.edu

***Abstract*—New products and small batch production entail a disproportionate amount of time is spent setting up automated assembly tasks. We have designed and implemented an automation tool to radically reduce this time. The setup for assembly automation involves: assembly planning, fixture-tool selection and positioning as well as part loading. This automation tool provides the robot with the ability to provide a human operator with precise and convenient instructions to follow through augmented reality, while at the same time allowing the robot to read information supplied by the human operator's actions. In this way, a complex setup task can be collaboratively executed, while allowing both the robot and the human to do what each does best.**

***Keywords*-Augmented Reality (AR), setup for assembly automation, Human-Robot Interaction (HRI), Robot-Human Interaction (RHI), active vision, multi-initiative tasks.**

I. Introduction

In today's modern manufacturing, there is an increasing demand for flexible, adaptable, yet efficient manufacturing systems. One of the key goals for achieving this is to ensure a rapid changeover between products, thus a reduction of the setup time is a necessity. Even in highly automated manufacturing systems, the setup of these processes are, for the most part, an expensive, time consuming and manual task performed by skilled operators.

The demand for costumer-product diversity along with the short product life-cycle has led to an increased focus on bringing the skills of human operators (flexibility, adaptability, etc.) together with the skills of robots (efficiency, repeatability, etc.). This emerging area of hybrid manufacturing systems has especially focused on Human-Robot Interaction (HRI) and Collaboration [1][2]. The work carried out in this area, covers a wide range of issues like effective teaching methods [3], ensuring human safety [4][5][6], communication [7][8], etc. However, there has been little focus on the initial setup of robotic work cells. We argue that this is an area that could also benefit from the principles of HRI, by using them as a tool for semi-automating parts of the setup process. Related to this we distinguish between systems where either the human has the initiative in the task (HRI) or the robot has the task initiative (Robot-Human Interaction (RHI)). This concept is foreshadowed in previous work [9][10].

In this paper, a concept for a new tool allowing for semi-automated multi-initiative setups is presented. This concept should provide a basis for improving the existing setup-processes by combining the precision of the robot with the judgment of a human operator.

The remainder of this paper is structured as follows: Section II discusses our approach to augmented reality applied to setups for robotic assembly. Section III illustrates the design and implementation of a new augmented reality tool, while section IV describes its general applications. Section V details two applications including guiding setups for robotic assemblies and part loading. Finally, sections VI and VII discuss future work and offer our conclusions.

II. Background

In a multi-agent collaborative system, one agent typically has the initiative. In other words, that agent is leading the task step-by-step by marking the beginning and end of a step with spoken or gestural commands. In a multi-initiative system, it is expected that the agents can trade the lead role based on which agent has superior knowledge or which agent has a superior vantage point for the task at hand. In this case, we are describing a system with one human and one robot agent, while both agents share a common goal of building a setup for the robotic assembly cell.

A. Human-Robot communication and collaboration

In general, the human agent has more complete knowledge about the general environment (e.g., lighting conditions, objects not modeled by the robotic agent or the moment-by-moment positions of human agents and managing their safety). In addition, there may be system goals that are not explicitly known by the robotic system (e.g., optimizing a machine component that is not modeled, giving priority to human safety or knowledge about delicate equipment and how it can be best safeguarded). For all of these reasons and others, the human agent may decide to take the task initiative. The robot on the other hand has it's own clear-cut position of superiority. For example, the robot can accurately model its own motions and do collision checking with other modeled elements. The robot can map its positions easily into world coordinates so that precise positions can be displayed for the benefit of a cooperative human agent and in many cases the robot simply can ”see” a developing situation better from its unique vantage point. Based on the give and take of the specific task elements, it is appropriate to allow the initiative to easily change hands between the agents.

To allow for mixed-initiative collaboration between a human and a robot agent respectively, it is necessary to ensure that they reach a common understanding (i.e., ”grounding”), easily and confidently. In previous work this area has been investigated based on human-human interaction, where both audio, visual, and environmental cues, respectively, are used [1]. This indicates that a single channel of communication (e.g., speech) is insufficient, and a multi-channel approach (as seen in [4]) to human-robot communication and collaboration is necessary. Ideally, this means that the robot must have the capability to both understand and respond across the available communication channels. This work focuses on providing the robot with the ability to understand and respond to environmental cues in the workspace. The robot communicates by reading the environment and human intent by means of active vision, while the robot can send information to the human through Augmented Reality (AR).

B. Augmented Reality as a tool for communication

In an assembly task (or setting up for an assembly task) we must choose a communication mode most suitable to both the robot and human agent. This communication mode could be speech [10] as it often is between human agents, but the bandwidth for this modality is very low and noisy, along with the ”vocabulary problem” [8]. Instead we have conceived a system that can both easily display and ”see” visual information. To accomplish this, we have designed a tool, seen on Figure 1, that can augment reality with complex laser displays and at the same time capture these visual images. With this tool, laser-displays can be registered to the real world so that the projective displays can provide precise ”pointing data” as well as embedded information.

With augmented reality we are able to ”enhance the real world” by merging virtual objects with real objects. An example of this is seen in [11], where a projector-camera system is used to display information directly onto objects in an unconstrained environment. Implementing this kind of functionality into a robotic work cell has the advantage that the operator is able to maintain focus on the working environment while receiving information and instructions visually. Previous work has also shown that assembly instructions given to an untrained operator through AR result in a faster execution with fewer errors [12].

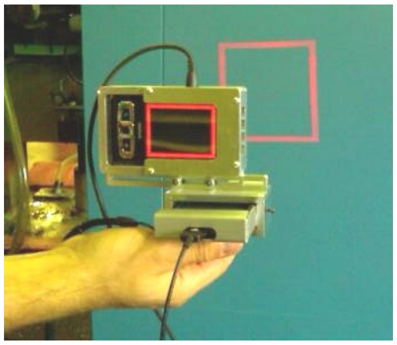


Figure 1. The augmented reality tool composed of a small laser-projector and a smartphone with a camera in a special fixture.

C. Active vision

To simplify the task of computer vision, the projected data can provide a way to see the structure of the environment via active vision [13]. The main idea behind active vision is that a simple, very bright pattern can be projected and then captured in an image. With simple image processing (e.g., thresholding followed by finding contours in the resulting binary image) it is possible to recover the distorted projections. Further, by analyzing the distortions of each projection, it is possible to directly infer the shapes of the 3D objects. In many ways, this is the holy grail of computer vision, but in the past the projection-setup has been bulky and awkward to use except for in the most constrained applications (e.g., single part inspection).

III. Hardware and Setup

The robotic hardware is as follows: an ABB IRB-140 6 axis robotic arm equipped with an Applied Robotics Smartgripper^{TM} electric gripper, a steel table as a foundation for varied configurations of fixtures (e.g., vise. clamps and compliant platforms), sensors (e.g., loadcells) and tools, which can be attached to the table with magnetic feet.

The augmented reality tool is constructed from: a Nokia N95 8G smartphone, a Microvision ShowWX Pico-P projector and a special fixture mount. The specific hardware used in this setup is just one of several possibilities, as other choices of hardware may be used. The smartphone requires several special features (i) programmable graphics (e.g., OpenGL used in this case) with, (ii) video output, (iii) wireless connectivity (e.g., Bluetooth and WiFi) and (iv) a camera with wireless video streaming. The projector needs to be laser based so that it is focus free and only the graphics are directly illuminated. The smartphone in turn streams data to a high-end PC suitable for doing real time computer vision and robot control.

There are two identical augmented reality tools situated at two distinct stations. The first station, seen in Figure 2, is attached to the robot gripper so that the robot can (i) use it as an intelligent pointer composed of a graphics window and (ii) use it for varied active vision tasks by pointing the camera at points of interest. The second station, seen in Figure 3, is fixed and can be used to analyze the robot, it’s gripper and the parts being held. In addition, there are tasks

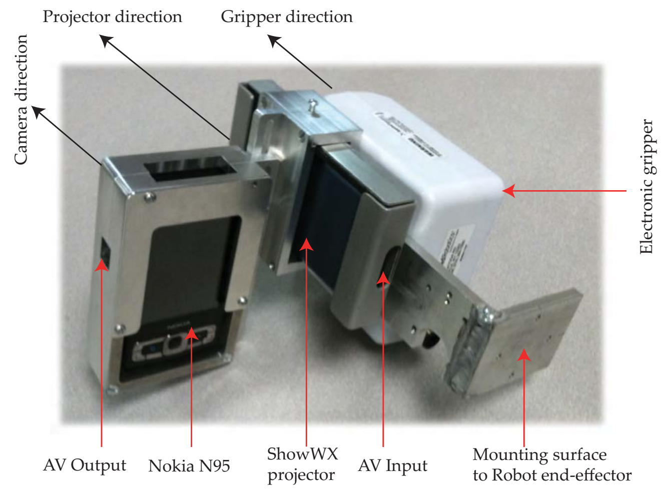


Figure 2. The AR tool assembled with the electric gripper for mounting on the robot end-effector.

that depend on projecting from one vantage point, while imaging from another (e.g., tasks involving triangulation) so for these tasks both stations work together.

There is a data command loop that operates between the workstation controller and the augmented reality tool. The typical steps for this command cycle are as follows:

1) *Workstation: Send* a high-level graphics command to the phone based on the application.
2) *Phone: Receive* draw command from the workstation.
3) *Phone: Draw* relevant pattern on phone's screen.
4) *Phone: Project* pattern on world.
5) *Phone: Snap* a picture of the resulting projection.
6) *Phone: Send* picture back to workstation.
7) *Workstation: Analyze* image and extract information.
8) *Workstation: Send* (optional) key information to phone - projector to be seen by human agent. This information can be registered on objects in the real world based on image analysis.

IV. Application areas for HRI in semi-automated setups

There are many application areas for automated assembly where both human and robot agents can effectively cooperate. These include (i) cooperative setups where many elements are not suited to robot handling (ii) human training of assembly methods and sequences (iii) robotic programming by demonstrations (iv) and success or failure determination in a given assembly step [14]. In an attempt to capture all of these possible applications, we have composed a general framework, illustrated in Figure 4. If needed, the human agent is able to perform some offline planning of robot tasks, environment, etc. These plans are then "mapped" to the robotic work cell where they are put into reality through the two AR stations described. Reversely, the robot may

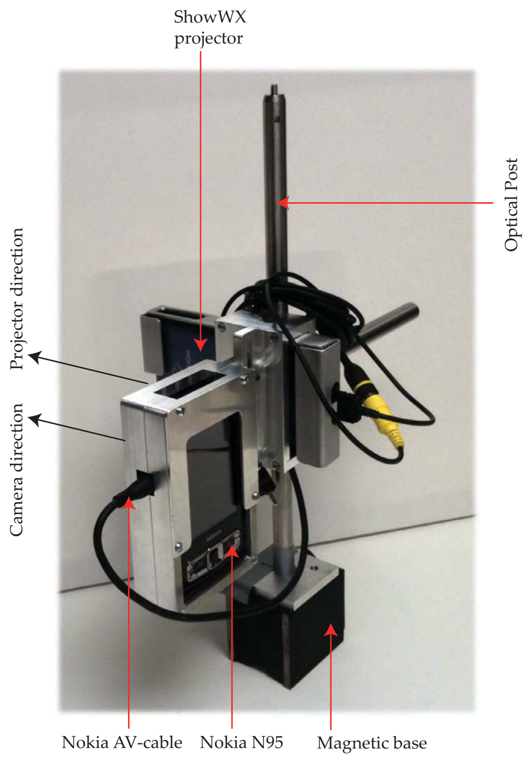


Figure 3. The AR tool mounted on a pair of optical posts for easy adjustments. The station can be conveniently placed by utilizing a magnetic base.

send information read from the environment through active vision back to the offline plan.

V. Detailed Applications

To demonstrate the general framework, we will present two applications that use multi-initiative execution of setups.

A. Positioning fixtures in robotic environment

The first application involves the precise positioning of magnetic feet on a steel table. These magnetic feet secure

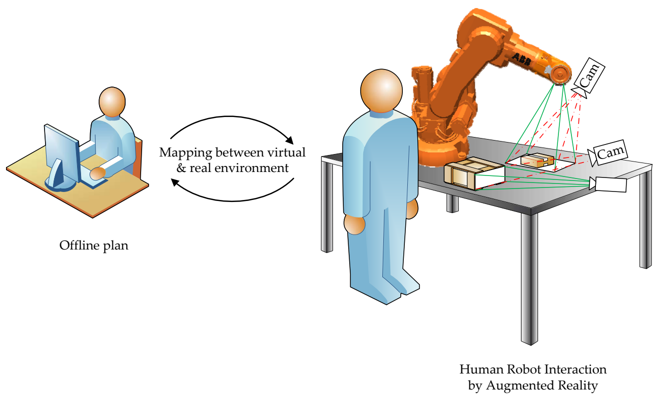

Mapping between virtual & real environment
Offline plan
Cam
Cam
Human Robot Interaction by Augmented Reality

Figure 4. A general concept sketch for using augmented reality in partially automated assembly.

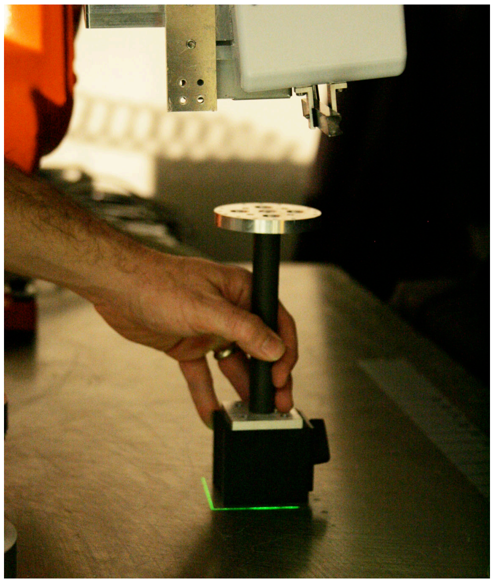


Figure 5. The AR tool marks a fixture's placement in green to aid the human operator.

a movable assembly platform on which assembled parts are held. In this application, planning software can locate an optimal location for the platform to avoid robot singularities and to provide collision free access to loading points. Since the robot controller knows the working coordinates and positions for all the fixturing elements, it can easily paint the environment with green laser marks to (i) request the human to load a fixture component and (ii) to provide a precise location. After the fixture elements have been positioned by the human-agent, the robot can then confirm the locations with the same system via active vision. The active vision algorithm projects light stripes at known locations and the fixture deforms the stripes into angles. The vertices of the angled laser stripes represent the transition between the surface of the table and the vertical wall of the magnetic foot. These positions can then be used to double check that the human operator has positioned the fixture correctly. Other researchers have shown [15] that more complex patterns (e.g., sinusoidal gradients) can be projected and in some cases it is possible to achieve better than sub-pixel resolutions, while reading object positions.

In this case, the robot has taken the initiative in the collaborative process of positioning fixtures. Alternatively, the human operator can ignore the robot's command and position the fixture in a fundamentally different place. In this case, the robot would have to discover the position and read the human's suggestion (again by active vision) of a better location. At this point, the planning system could change the planned fixture locations and run validity tests on that new set of locations. If the position is deemed feasible (e.g., no robot singularities prohibiting movement and collision free access to parts), then the plan could be altered in all of the specific details (i.e., part and fixture locations and robot motion plans). At this point, the robot would once again seize the task initiative and continue to give instructions of where additional fixtures need to be placed given the first position provided by the human. Thus, the time consuming process of relocating fixtures and other environmental components can be transformed into two phases: (i) initial conception of new approach/solution by human-agent and (ii) delegating the tedium of completing the task to the robotic agent.

B. Loading parts in fixtures and robot gripper

The second application helps the human-agent to precisely load a part into the robotic gripper during development to test, optimize, and/or train an assembly strategy. This entails that the operator is provided with information about (i) which part to feed to the robot and (ii) the orientation and location of the part in the robotic gripper. The second piece of information is especially important since the placement of the part might offer multiple solutions. For instance, the operator might choose to place the part upside-down in relation to the assembly strategy, causing the assembly operation to fail. In addition, the operator needs to place the parts at the same position in consecutive runs.

In this application, the information is provided by digitally marking the robotic gripper through AR by projection. The placement of the part is found by planning software, and then fed to the robot and an AR tool. By placing the robotic gripper in the AR tool's field of view, the tool detects the location of the gripper fingers through active vision and then projects a digital stripe marking the proper location of the part.

To properly detect the gripper fingers, successfully calculate the stripe location, and finally project this line onto the finger, it is necessary to find a relation between three

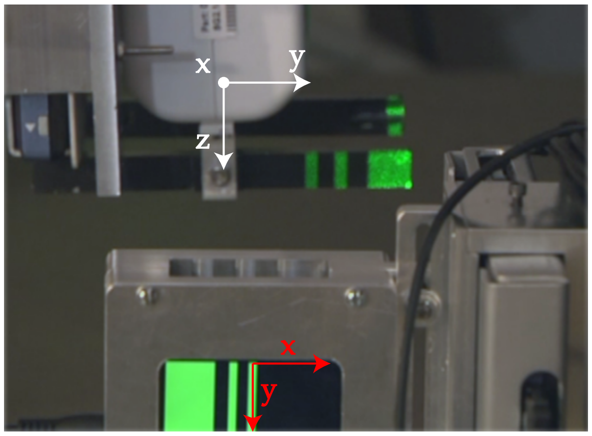


Figure 6. The robot and the projector coordinate frames are two out of the three frames that need to be correlated. The third coordinate frame is the incoming camera image seen in figure 8.

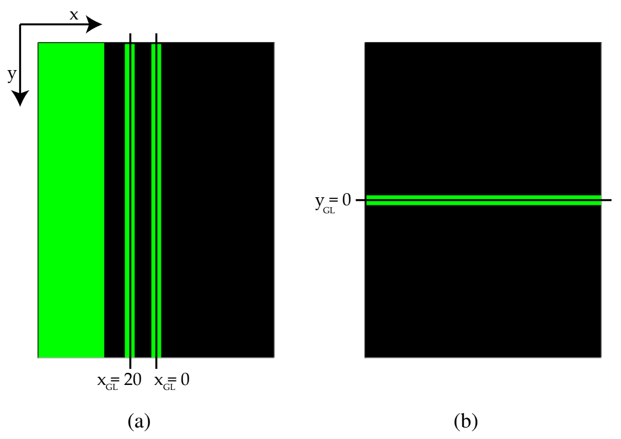


Figure 7. These patterns are used to detect the gripper fingers and to correlate the three coordinate frames.

coordinate frames: (i) The robot frame and (ii) the projector image frame, as illustrated in Figure 6, and finally (iii) the camera image frame as seen Figure 8. To find this relation we utilize the patterns shown in Figure 7.

The relation between the camera image and projector image coordinate frames is found by relating the known pixel positions of the thin lines of the pattern in Figure 7(a) to the distance between the corresponding lines found in the camera image seen in Figure 8. Note that the projected pattern, seen in Figure 8, is rotated by 180 degrees as a result of the projector orientation. The relation to the robot coordinate frame is found by moving the gripper fingers into the path of the projected patterns in the horizontal (Figure 7(a)) and vertical (Figure 7(b)) directions respectively, until they are successfully detected in the camera image. Thus, the complete relation between the three coordinate frames are found in the following steps:

1) The pattern shown in Figure 7(a) is projected.
2) The robotic gripper is moved horizontally into the path of the projected pattern until the full pattern is recognized (as shown in Figure 8) in the resulting camera image.
3) The pattern is changed to the one shown in Figure 7(b), and step 2 is repeated in the vertical direction.
4) The resulting robot position is saved, and the stripe is calculated from two captured images showing the two projected patterns respectively, yielding the result shown in Figure 8.
5) A line is created based on the calculation and projected onto the gripper (left-most stripe in Figure 8).

Once the stripe has been successfully calculated and projected, the operator is able to load the part into the robotic gripper at the marked position.

In this application, the robot also takes the initiative, telling the human agent which part to load, and how to place it. As previously discussed, it might be possible for

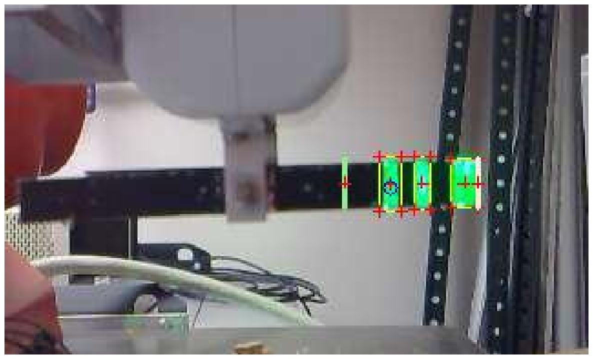


Figure 8. The result after image processing, which shows the calculated green stripe (left most) for part placement.

the operator to load the part in several different orientations and locations. This raises the possibility that the operator misinterprets the information, or choose to ignore it, and as a result loads the part incorrectly. Thus, the robot needs to have the ability to check that the part has been loaded correctly, and perform a suitable correcting action in case it is not. This could involve requesting the operator to correct the mistake, or, if possible, change the motion plan according to the loaded configuration of the part.

This application highlights a new design component for this approach; namely, we often must design different projected patterns for different applications. In this application, a 3 stripe pattern was chosen to simplify visual identification of the green stripes and its subsequent processing to determine positional information along the gripper fingers. We also have experimented with the mixing of primary colors to compose different information in the same space during projection. Once visual processing begins, the color channels can be separated and processed individually. The primary advantage of this approach is to allow one picture to encode several different messages simultaneously, while noting that sending pictures from the AR tool back to the workstation is a key processing bottleneck. Other well known patterns include a checkerboard pattern to calibrate the projection-camera system and sinusoidal wave patterns to achieve sub-pixel point cloud reconstructions. We fully expect that other new applications will demand new and creative projections to help manage and minimize the effect of characteristic limitations found in these applications.

VI. Future work

We have discussed several applications with relatively simple steps performed by the human and robot agents. As the applications become more complex, it will be necessary to have dialog markers to determine when one step in an application ends and the next step begins. To seamlessly switch initiative between agents also requires clear dialog markers to avoid confusion or a time consuming end-of-step confirmation process. For example, in our first application we described a system where a person is directed to place a fixture at a given location, but they may choose to place it someplace entirely different. In that case, the robot would have to attempt to confirm the given placement and explicitly fail. This failure mode in itself is the dialog marker that allows the systems to swap initiative. However, this is not the most efficient method. One option might be to provide a simple visual marker (or coin) painted red on one side and green on the other. By flipping the coin, the human operator can either seize or relinquish the initiative. Of course, it is possible to do this process digitally (computer input), but that may require accessing a teach pendant or computer interface that would be relatively inconvenient. In addition, humans find speech to be the modality of choice especially when only dialog markers are required [7]. In the end, we need to find a method that does not impede progress in a given task and is more resistant to background noise.

The augmented reality tool we have described in this paper is ideal for tasks that require a narrow field of view. Unfortunately, that fails to address many applications that require a wide-field of view with less precision. For example, if we wished to monitor the movements of the human operators for safety analysis, then a different device such as the new Microsoft Kinect would be required.

Finally to achieve acceptance in the world of manufacturing, it is necessary to do a carefully controlled study comparing times and costs between these methods and the state-of-the-art.

VII. Conclusion

The rapid miniaturization of key components: full color laser projection, wireless computing and video streaming will make a wide array of augmented reality tasks feasible. By adding these functions to automated robotic systems, it also becomes apparent how humans and robots can collaborate on complex tasks without excessive programming overhead. This is particularly important in manufacturing where the time to setup machines is becoming a dominant bottleneck in production due to smaller and smaller batches and shorter product life-cycles.

Acknowledgment

We would like to thank Michael Dawson-Haggerty for his development work to guide the placement of fixtures. We would also like to thank Otto Moensteds Fond, Knud Hoejgaards Fond, Oticon Fonden, Vilhelm Pedersen & Hustrus Legat and Aalborg Stiftidenes Fond for the financial support for Dan Gadensgaard during the period of this project.

Finally, we would like to thank ABB Corporate Research for their partial support of the entire project. Thomas Fuhlbrigge (ABB) and Gregory Rossano (ABB) both provided significant feedback during the work.

References

[1] S. A. Green, M. Billinghurst, X. Chen and J. Chase, ”Human-Robot Collaboration: A literature review and Augmented Reality approach in design,” *International Journal of Advanced Robotic Systems*, Volume 5, 2008, pp. 1-18.

[2] J. Krüger, T. K. Lien, and A. Verl, ”Cooperation of human and machines in assembly lines,” *CIRP Annuals - Manufacturing Technology*, Volume 58, 2009, pp. 628-646.

[3] F. Walhoff, J. Blume, A. Bannat, W. Rösel, C. Lenz, and A. Knoll, ”A skill based approach towards hybrid assembly,” *Advanced Engineering Informatics*, Volume 24, 2010, pp. 329-339.

[4] M. Zaeh and W. Roesel, ”Safety aspects in a Human-Robot Interaction scenario: A human worker is co-operating with an industrial robot,” in *Progress in Robotics*, Springer Berlin Heidelberg, 2009, pp. 53-62.

[5] J. A. Corrales, F. A. Candelas, and F. Torres, ”Safe Human-Robot Interaction based on dynamic sphere-swept line bounding volumes,” *Robotics and Computer-Integrated Manufacturing*, Volume 27, 2011, pp. 177-187.

[6] N. Lauzier and C. Gosselin, ”3-DOF cartesian force limiting device based on the delta architechture for safe physical Human-Robot Interaction,” in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 3-8 2010, Anchorage, Alaska, USA, pp. 3420-3425.

[7] M. Lohse (2011, Jan. 6), ”The role of expectations in HRI,” in *AISB'08 Workshop on New Frontiers in Human-Robot Interaction*, 2009, Edinburgh, UK [Online]. Available: http://www.aisb.org.uk/convention/aisb09/Proceedings-/NEWFRONTIERS/FILES/LohseM.pdf

[8] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais, ”The vocabulary problem in human-system communication,” *Commun. ACM*, Volume 30, 1987, pp. 964-971.

[9] J. Peltason, F. H. K. Siepmann, T. P. Spexard, B. Wrede, M. Hanheide, and E. A. Topp, ”Mixed-Initiative in human augmented mapping,” in *IEEE International Conference on Robotics and Automation (ICRA 09)*, May 12-17 2009, Kobe, Japan, pp. 2146 - 2153.

[10] I. Lütkebohle, J. Peltason, R. Haschke, B. Wrede, and S. Wachsmuth (2011, Jan. 8), ”The curious robot learns grasping in multi-modal interaction,” in *IEEE International Conference on Robotics and Automation (ICRA 2010)*, May 3-8, Anchorage, Alaska [Online]. Available: http://aiweb.techfak.uni-bielefeld.de/files/cr-video.pdf

[11] D. Molyneaux, H. Gellersen, G. Kortuem, and B. Schiele, ”Cooperative augmentation of smart objects with projector-camera systems,” *UbiComp 2007: Ubiquitous Computing*, Springer Berlin / Heidelberg, 2007, pp. 501-518.

[12] A. Tang, C. Owen, F. Biocca, and W. M. Mou, ”Performance evaluation of augmented reality for direct assembly,” in *Virtual and Augmented Reality Applications in Manufacturing*, Springer, 2004, pp. 311-331.

[13] B. Zhang, E. J. Gonzalez-Galvan, J. Batsche, and S. Skaar, ”Computer Vision” published by I-Tech, Vienna Austria, November 2008, pp. 111-123.

[14] A. Rodriguez, D. Bourne, M. Mason, G. Rossano, and J. Wang, ”Failure detection in assembly: Force signature analysis,” in *IEEE Conference on Automation Science and Engineering (CASE 2010)*, August 21-24, Toronto Canada, pp. 210-215.

[15] T. Peng and S. K. Gupta, ”Model and algorithms for point cloud construction using digital projection patterns”, *Transactions of ASME*, Volume 7, 2007, pp. 372-381.

A Distributed E-health Model Using Mobile Agents

Ali A. Pouyan
School of Computer and IT Engineering
Shahrood University of Technology
Shahrood, Iran
e-mail: apoyan@shahroodut.ac.ir

Sadegh Ekrami
School of Computer and IT Engineering
Shahrood University of Technology
Shahrood, Iran
e-mail: s.ekrami@comp.tus.ac.ir

Momeneh Taban
School of Computer and IT Engineering
Shahrood University of Technology
Shahrood, Iran
e-mail: m.taban@comp.tus.ac.ir

***Abstract*—In this paper, a 3-layer agent-based model is proposed. The proposed model is supposed to be used as a general framework for e-health systems to facilitate health care processes. This 3-layer model consists of patient layer, clinic layer and central-hospital layer. These layers are hierarchically and horizontally related together. This relationship is defined in order to establish more accuracy in treatment recognition, quick treatment, and diagnosis. It facilitates the relationship between different parts of e-health procedure and reduces the workload and complexity to connect different interfaces. Mobile Agents technology is used to distribute the processing load as well as to support a more flexible peer-to-peer model, scalability and decentralization of control.**

Keywords–distributed; mobile agent; modeling; e-health; layer; workload.

I. INTRODUCTION

E-activities have been emerged as potential prospective trend in almost all aspects of human life. Some of them have become properly essential to control the huge amount of data and information produced during daily activities. At the same time, the introduction of e-Health represents the commitment of information and communication technologies to improve healthcare systems [1].

The E-health care indicates recording, measuring, monitoring, managing and finally delivering patient-oriented and specific-condition care services through Internet at the real-time.

E-health provides the interaction between a patient and healthcare institute, as well as institute-to-institute transmission of data or peer-to-peer communication between patients or doctors. Nowadays most healthcare institutes use computer systems for record keeping. Most of data being recorded would not be easy to understand for average patients, and they can be solely used directly by doctors and other practitioners. On-line tools may help patients to better understand their condition. Furthermore, it is of a great importance for a healthcare institute to manage, maintain, collect, and analyze the data received by doctors, practitioner and patients.

Several approaches have been proposed to integrate distributed information sources in a healthcare [2]. In one approach [3], the focus was on providing different teams across several hospitals with management assistance by coordinating their access to distributed information. The brokering architecture is centralized around a mediator agent which is responsible for allocating an appropriate medical team to an available operating theatre in which the transplant operation may be performed.

In this paper, the analysis is done across distributed system according to hierarchical and horizontal connections among the patients, the healthcare clinic, and the central hospital. We illustrate some benefits of these connections which lead to reduce the complexity of collaboration between some global queries and management of local entities. In order to have accurate queries and efficient analyses from distributed databases, a framework is required in order to support interoperability mechanism to identify and provide measures necessary to deal with the differences in database models. This makes use of a knowledge-based approach for information retrieval which can be provided by defining the entities and their relationships as well as agents' role.

The rest of this paper is organized as follows: Section 2 describes the agents reducing the complexity of the communication and facilitating the interaction between the layers in distributed environment. Section 3 explains the roles of multi agent system (MAS) in three layers and its efficiency in operating between layers, and also the interaction between layers with emphasizing the role of Object Request Broker (ORB) for being applied in client-server architecture. At the end of this paper, the conclusion and future work are presented.

II. MULTI-AGENT SYSTEM

A multi-agent system is a paradigm for understanding and modeling distributed systems, in which it is assumed that the computational components are autonomous; that is they are able to control their own behavior in the furtherance of their own goals. During the past decade, several agent architectures have been proposed to implement agent-based systems, and also a few efforts have been made to formally specify agent behaviors [4].

However, there is little research on narrowing the gap between agent formal models and agent implementation; and of course this paper will not focus on implementation phase.

Because of their inherent characteristics, such as code and state mobility, network awareness, and intelligence, Mobile Agents are able to enhance distributed applications. Some of them include reduction of the network bandwidth use, distribution of processing, support for a more flexible peer-to-peer model, scalability and decentralization of control. In terms of processing and network bandwidth consumption, the use of mobile agent paradigm is justified when use of some remote resource applying traditional approaches as client/server paradigm are more expensive than use of the agent.

The integration of many clinical activities in e-health, requires a wide area of e-health monitoring in which most of the patients are usually mobile and situated in a low bandwidth, high latency, asynchronous transaction and unstable connection communication environment while servers are highly distributed and heterogeneous. For this paradigm Mobile Agent technology is more suitable as it provides a powerful and efficient mechanism to develop application for a distributed and heterogeneous environment. Mobile agent technology offers the possibility to execute duties in an automated way with minimal human intervention. It allows medical staff to concentrate their attention on other activities and consequently save valuable time of medical resources. Also, the medical team can be always accessible and their decisions are easily monitored by medical staff and Central Hospital.

The mobile agent in each machine interacts with stationary environment (e.g., service agents or other resources) to accomplish its task. The Central, the clinic, and the patient's PC are three stationary agencies which are supposed to provide an environment for the agents to interact in the system.

A principal goal is to overcome the difficulties of coordination and communication. The main challenge is to identify types of agents, their tasks, goals and interfaces. The types of the agents which are used in the whole system are presented in Table I.

In fact, there is a large amount of patients' data stored in lots of databases and not translated into any knowledge form which can be effectively suitable for consumers whether they are patients or medical practitioners. To customize this health information, they are required to be retrieved and translated into a suitable knowledge. In this model Analyzer Agent is supposed to perform such a task to retrieve appropriate information and convert it into knowledge-based information.

TABLE I. AGENT'S GOALS AND TASKS.

Agent type	Goals	Tasks	interfaces
Profile Agent	Complete patient's information. Update patient's health information.	Record user's application. Record user's preference. Record user's tasks.	Virtual medical team
Advice Agent	Care improvement	Bring medical suggestion to the patient	Patient
Query Agent	Obtain statistical information. Collect information from distributed DBs.	Carry and run a request through the system.	Clinic - Central
Analyzer Agent	Obtain an appropriate request and analyze data and convert it into appropriate information	Analyze data from information resource	Research team - Patient

There is a main resource in central hospital databases which is consisted of any necessary information about maladies, their drug's side effects, statistical information, etc. It will be easier for the agents to access the information and compare them with the decision made by e-health team; even in case of any collision or mistake, the system would alert and make the team aware.

A patient who decides to join electronic healthcare clinics will be continuously inspected by a virtual team. Each user is represented by a mobile agent, thus the virtual medical team is always available. Virtual team may consist of doctors, nurses, practitioners and any department (e.g., laboratory, radiography, etc.) being able to make a connection locally or globally (depending on their authority and their connectivity).

In fact, there is a dynamic virtual team which has been formed to response needs of any particular patient in any point of time. The patient will be assigned to the appropriate medical team by its Profile Agent. The Profile Agent representing the description of a patient, his situation and his health signs, will be sent manually or automatically to the healthcare clinic. At the clinic side, there must be a team or a doctor to advise the patient according to the received condition. Once the Profile Agent is received by clinic, the clinic agency verifies the authenticity of the received data to detect which parts are new or maybe changed. Thus it is able to update the information and dispatch necessary parts to the virtual team members. This procedure is shown in the Fig. 1.

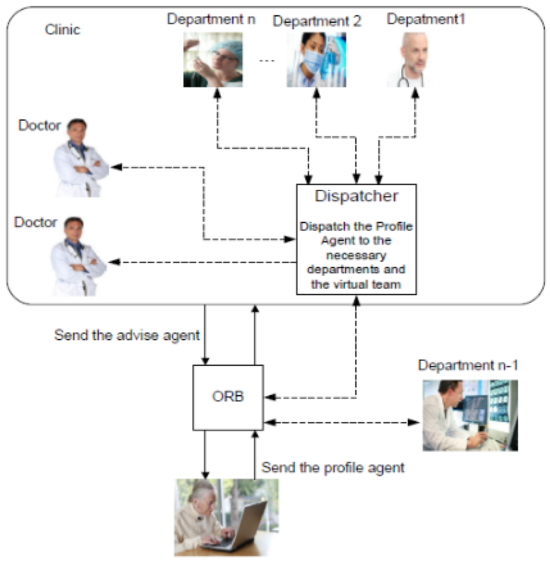


Figure 1. The patient's communication to the Clinic, its virtual team, and related department like laboratory, radiography, etc.

In this case, different conditions will be supposed depending on the structure of the clinic and their Network Structure. For example, the sub-sections of the clinic including laboratory, radiography, etc. are connected to the same authority (their connection are locally authorized) or maybe it is needed to connect other clinics with help of a broker to connect the entity A (from one clinic) to the entity B (another clinic).

The system is able to bring the team members together even if they are separate of each other. As mentioned earlier, some agents are defined to do system's tasks. Some of them are intelligent and able to analyze and verify the patient's information or team advices (As it has been mentioned in Table I).

Agents learn how to simplify choices according to pre-defined data rules and user-based information [5]. As a matter of fact, the rate of human errors in this area is not negligible; for instance in order to be sure of drugs consumption or their side effects, the patient can send an Agent to receive additional information on that special case. In fact, the patient does not know how it works. The only thing that the patient deals with is the interface and the result. This scenario promotes the use of mobile agent in order to decrease network usage.

This system could be used for different reasons including long-term care, homecare, clinical search studies and public health reporting [6].

III. LAYERED-BASED MODELING

In this paper, 3 layers are defined interacting with each other: the patient layer, the clinic layer and the central hospital layer. These layers are hierarchically and horizontally connected together. This relationship between layers is defined in order to establish more accuracy of treatment recognition, quick patients' treatment, and malady recognition, and also to facilitate the relationship between different parts of e-health procedure. As a scenario, both connections may be verified in details. If a patient is under cares of e-health, the doctor will visit the patient and it can be called a hierarchical connection. The receptor fills referral form and then sends it to the clinic's database (DB) and the latter will organize a new record for the patient. This is the beginning point of the patient - healthcare clinic interactions. After that, depending on patient's situation, an appropriate e-health team will be organized. At the same time, a horizontal connection between clinics or departments can be set in order to allocate the e-health team for the patient. A horizontal connection shows the same range of information interaction which is accessible for whole e-health team. Patient's information for each part of the virtual team appears in a specified interface. According to the responsibilities assigned to each section, some information or prescriptions are added to DB and they are shared with other parts to be more observed. Horizontal connection plays an important role for e-health team. Quick access, easy connection and quick changes of information by e-health team is the major role of a horizontal connection and it can lead e-health team to have a personalized application because in spite of being strongly related to the structure, it does not have any relation to the interface.

The access of each part of e-health team to the patient's information does not have any overlapping and each part can add its information when it is allowed to do.

A specific software is embedded on both sides (patients and e-health team) with two different interfaces. On the patient side, the patient or some tool such as an electronic health device or a portable device being able to interact with the patient's PC can be applied to control and measure some health parameters. The embedded software can analyze the data prepared at patient side and then the result will be sent as a Profile Agent which transmits health parameters of the patient to the clinic.

The interaction between client and server with different interfaces, program languages, hardware platforms, Operating Systems, location, etc. poses a great concern that how we can encounter with all of these parameters in the server side. A programming model acting as a middleware between clients and servers is needed. Object Request Broker (ORB) is a middleware from which a client can request a service without knowing anything about the servers which are present on the network. ORB provides the participating system with a consistent interface by wrapping each software entity in the framework in order to enable them to collaborate [7]. Various ORBs receive the requests, forward them to the appropriate servers, and then give the results back to the client. The structure of the ORB as well as the relationships of the system is shown in Fig. 2.

Interoperability means the ability of diverse systems to work together accurately. They must be able to exchange data and information in an effective, accurate and continuous way. Core problems with which e-health system faces are lack of interoperability, common standards and common architecture for their health information exchange.

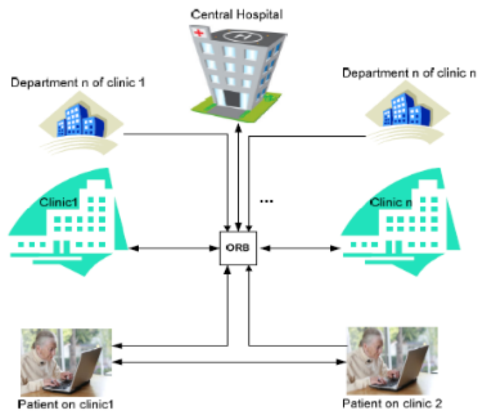


Figure 2. Role of ORB and structure of whole system

Since public health agencies rely on multiple systems, it is a hard task to provide a common framework for all relevant parties. Moreover, a middleware is required to connect different sides in order to omit dependency on various applications and different architectures of information exchange.

Although common standards such as Logical Observation Identifiers Names and Codes (LOINC), Health Level Seven International (HL7) have been recognized as long term important factors, at present time it is unknown how they shall be implemented. Moreover, their implementation needs huge amount of funding. So beside of this action, this paper suggests a middleware called ORB in order to establish a connection between client side, server side, and other different sections in different levels of authorities across healthcare monitoring team, departments, knowledge-based resources, etc.

Up to now, the system has established a hierarchical connection from the patient side (client side) to the healthcare team (server side). Diagnostic team sends some request to the main resources of the central hospital in order to obtain information from the knowledge-based data server. The analyze agent is responsible to obtain necessary information and convert it into appropriate knowledge-based information all are to be transferred by the ORB. Sometimes the doctor wants to know the patient's health information, so the system creates an Agent to retrieve the appropriate information from different parts such as radiology or laboratories; therefore, the doctor will not conflict with these complexities of the system and its procedures. There are other facilities on this system that allow the patients to communicate directly with each other. In fact the system can find some patients which are engaged in the same diseases. This can be a mental help for the patients and their relatives.

A) Patients

The patient or the electronic health care tools import data on the patient side interface and submit it by producing an agent containing the data and its respective function to have more interaction with the clinic. In the next step, this data is sent to the ORB. The ORB recognizes the agent type and invokes the appropriate function to save the data in the related clinic's DB. The data is detected by practitioners or medical teams on their screen and they diagnose the patient's status. If they want to prescribe the patient, the system will make an Advice Agent which will be sent to the patient's PC.

The patient's PC generates different agents to perform the tasks, like Profile Agent as described above or Query Agent that the patient uses it to access the medical information collected by Analyzer Agent. To make sure of the medical process and getting some reports, the system needs to make Query Agent and make use of pre-defined knowledge to analyze and send back the result.

The Analyzer Agent can update its information using basic information, the result of a patient's care, advices of doctors or some new information which has been added to the knowledge-based resources.

B) Healthcare Clinic

Once a patient connects to the e-health system, he/she will be assigned to appropriate departments and be advised by a virtual team. The virtual team may need to work separately instead of being together in a place.

The Profile Agent consists of different parts which must be divided in order to be analyzed in different departments and to be observed by a virtual team. The dispatcher is responsible to check Profile Agent and distinguish each section. As the virtual team and departments are not necessarily in the same place, the dispatcher must use two different connections to transfer appropriate data to appropriate department, local connection and global connection. Global connection refers to a connection established with the help of the ORB, while local connection does not need the ORB for applying the connection.

Clinics can use the other clinics' information via the ORB. For the sake of security and safety, there should be some different accessibility for the personnel who want to achieve this information. In this procedure, the clinic will introduce itself to the ORB and also define its role. The ORB will verify the Agent received from the first clinic and also define the accessibility of the request depending on its role. Then it will negotiate with the second clinic to verify the permission of accessibility. If the second clinic confirms the accessibility, appropriate information will be sent back to the first one through Query Agent.

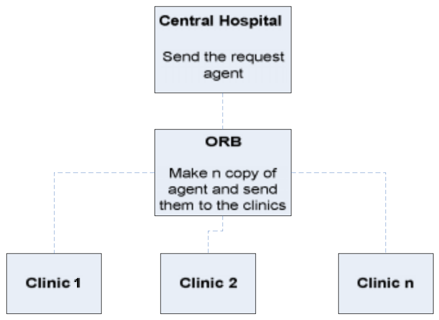


Figure 3. Role of Query Agent and Analyzer Agent to analyze the information received from Clinics.

C) Central

In a periodic sequence, central hospital makes a Query Agent to retrieve information of clinics' DBs. The central hospital can analyze this data and convert it into statistical information usable by research teams; for example, medical diagnoses, procedures, allergy information, medication, patient histories, eligibility data, and so on.

The central will analyze the patient's data and the clinic's one; the final output is statistical information. This knowledge-based information is useful for the clinics and research teams. Analyzing the patient status, based on the illness, amount of drugs' consumption, age, data gained from some tools or patients itself, etc. will be useful for the healthcare clinics to achieve some new medicine research and also to provide better patients' care.

It is necessary for the system to know which user is on-line or off-line and where they are. The central also analyzes and monitors the queries and the functions which are being interacted among the users and the healthcare clinics through the ORB. A global observation mechanism is needed to monitor and analyze the status of these entities [7].

Fig. 4 explains the functionality of the Query Agent and the Analyzer Agent in pseudo codes. In fact the role of Query Agent and Analyzer Agent will be detected according to their interoperability and their functionality.

The Query function can be divided into different Query functions. It means that a query may want to retrieve information from many resources or make another query from another resource by receiving especial information. The whole information will save in Result. After that the Analyzer Agent processes the data which has been received by many resources. In fact it will compare the information with the knowledge of the main resource, and it can report the appropriate information to the requester.

```
QueryAgent()
{
make child Query (variable type, query type); /* "variable type" is the type of some information parameters required to obtain the appropriate information for example what are the causes of the X disease.*/
Result=Query (variable type, query type); /* Result of the query according to its purpose is reserved in "Result" */
if (Analyzer==true) {
Result=AnalyzerAgent(Result); /* the Query Agent can invoke the Analyzer Agent – the Analyzer Agent can analyze the information which has been taken by the Query Agent from different part such as laboratory, patients, doctors, etc. */
Return Result;
}
}
AnalyzerAgent()
{
Info=Obtain (variable type); /* "variable type" is the type of some information parameters required to obtain the appropriate information.*/

Check if (variable type is reported before) {
Info=Obtain (title); /* the necessary title which are more related according to their subject and titles */
}
Result=Compare (Info with pre-defined knowledge); /* the obtained information with pre-defined knowledge and make some statistical report in a file */
Save (variable type);
Save (Result);
Return Result;
Pre-defined knowledge:
According to the variable type, will detect some necessary information around the title of the query and some related titles (it can make use of previous queries which were used before and will compare all of them)
}
```

Figure 4. Pseudo code of the Query Agent and the Analyzer Agent

IV. CONCLUSION AND FUTURE WORK

This paper introduced a framework for an e-health system to facilitate health care processes. A layered-based system and ORB are used to reduce the workload and the complexity of the connection among different interfaces. Mobile Agent is used for distribution of processing, support for a more flexible peer-to-peer model, scalability and decentralization of control. Generally speaking, there are some specific applications which run on this platform:

- Health records
- Telemedicine
- Tele monitoring
- Agent-based transition of data
- Decision support
- User identification
- Information accessibility in mobile state

Moreover, a common set of rules and standards should be established for information exchange, interoperability and making sure of data quality which could be a part of our anticipated research. Our future work will focus on analyzing the protocols between layers, adding some intelligent agents to facilitate some tasks and more about the security of agent implementation.

REFERENCES

[1] M. Eichelberg, T. Aden, and J. O. Riesmeier "The Promise of e-Health-a Canadian Perspective", e-Health Int, 17(1), Sept 2002. pp. 1-47.

[2] V. Shankaraman, V. Amorosiadou, and B. Robinson, "Agents in Medical Informatics", in Proc. Of IASTED International Conference on Applied Informatics, Austria, 2000. pp. 489-493.

[3] A. Moreno, A. Valls, and J. Bocio, "Management of Hospital Teams for Organ Transplants Using Multi-Agent Systems", Artificial Intelligence in Medicine, Lecture Notes in Computer Science, Springer Verlag, pp. 413-416, 2001.

[4] D. Xu, J. Yin, Y. Deng, and J. Ding "A Formal Architectural Model for Logical Agent Mobility" IEEE Transactions on Software Engineering, Vol. 29, No. 1, January 2003, pp. 31 – 45.

[5] I. Jung, D. Thapa, and G. Wang. "Intelligent Agent-Based Graphic User Interface (GUI) for e-Physician". World Academy of Science, Engineering and Technology 36, 2007, pp. 442-749.

[6] A. Pitsillides, G. Samaras, B. Pitsillides, D. Deorgiades, P. Andreou, and E. Christodoulou. "Virtual Collaborative Teams for Home Healthcare". Journal of Mobile Multimedia, Vol. 2, No.1 (2006) 023-036, pp. 23 – 36.

[7] K.-M. Chao, R. Anane, J. Plumley, N. Godwin1, and R.N.G. Naguib "A Mobile Agent Framework for Telecardiology", Engineering in Medicine and Biology Society, 2001. Proceedings of the 23rd Annual International Conference of the IEEE Volume 4, Issue, 2001, pp. 3484 – 3487, vol.4

[8] F. Cao, N. Archer, and S. Poehlman, "An Agent-based Knowledge Management Framework for Electronic Health Record Interoperability". *Journal of Emerging Technologies in Web Intelligence*, Vol 1, No 2 (2009), 119-128, Nov 2009, pp. 119-128.

[9] A.M. Masaud-Wahaishi and H. Ghenniwa. "Privacy-Based Information Brokering for Cooperative Distributed e-Health Systems". Journal of Emerging Technologies in Web Intelligence, Vol 1, No 2 (2009), pp. 161-171, Nov 2009.

[10] O. Ajayi, R.O. Sinnott, and A.Stell, (2008) "Dynamic Trust Negotiation for Flexible e-Health Collaborations*"*. In: Mardi Gras Conference 2008, 30 January - 2 February 2008, Baton Rouge, USA. pp. 165 – 172, Doi: 10.1145/1341811.1341821.

[11] O.B. Henkemans, S. Bonacina, N. Cappiello, Ch. van der Mast, M. Neerincx, and F. Pinciroli. "A Hybrid Multi-Agent System Architecture for Distributed Supervision of Chronic Patients in the e-Health Setting". Euromedia 2007, pp. 119-127, ISBN Number, 9789077381328.

Modular Structure of Neural Networks for Classification of Wooden Surfaces with PLC Industrial Implementation

Irina Topalova
Automation of Discrete Production Engineering
Technical University of Sofia
Sofia, Bulgaria
itopalova@tu-sofia.bg

Alexander Tzokev
Automation of Discrete Production Engineering
Technical University of Sofia
Sofia, Bulgaria
alextz@tu-sofia.bg

***Abstract*—This paper presents development and research results, applying new approaches and means to the design of Modular Structure of Neural Network for classification of wooden balks (MSNN). It is based on machine vision, unified recognition algorithm and modular neural network structure for real time operation in a standard Programmable Logic Controller (PLC). MSNN is modular, which provides possibility for constructing different structures using in parallel many neural network function blocks with different topologies. It includes development of decision making method for obtaining high recognition accuracy in texture classification using histograms as input data. The method simplicity combined with the modular performance contributes to fast computations and high flexibility of the proposed system. The modular MSNN, containing standard functional blocks, can find application in different applied science fields.**

Keywords – texture classification; machine vision; automation; PLC.

I. INTRODUCTION

The main disadvantage of the processes at the enterprises for production of high quality wooden items is the lack of objective quality control on the textures of the used wooden balks before any further treatment process. The identification of the type and quality of the balks is accomplished through subjective assessment, which leads to a lower material utilization factor and lower the quality of the produced items. There is a demand for automated visual control, which should include the recognition of defects within the whole range of produced surface structures and their classification. This will result in optimized cutting up and fitting of the produced parts. The international state-of-the-art shows that the company “Michael Weining-Technowood” Ltd, Germany [1] is the leading producer that has introduced in the woodwork a method for texture quality automated control applying optical scanning of the surface and cutting optimization with defects elimination. The firm has not developed automated selection of wood balks with different textures in the process of wedge-shaped splicing, as well as it offers complete technological equipment at a very high market price. In this case, the use of different visual sensors does not contribute to the unification of the used methods for recognition. The system is developed in accordance with the specific equipment, available at the company.

A. *Current state of similar approaches*

Most research in the wood product industry has been applied in the development of automatic visual inspection systems for the purpose of grading and edging based on the quality of the wood and the presence of defects. These technologies use devices such as ultrasound, microwave, laser ranging, cameras and spectrometers, which are rather expensive. Another wood classification system based on several multi - layered - perceptron (MLP) neural network models has been developed [2]. The MLP structure has been trained by the authors, using 20 input features (Angular Second Moment, Contrast, Correlation, Inverse Difference Moment, Entropy, etc., for five different image rotations), extracted from the texture. In this case, the authors have obtained 1sec overall computation time and 95% accuracy. The main difference considering our approach is the obtained 679 ms computation time and the implementation of the method in standard industrial PLC. Another method [3,4] for defect detection in textured wood surfaces relies on the analysis and fusion of image series with variable illumination. This method can be considered as filter-based, where the filters or feature detectors are learned from a set of training surfaces. It spends 0.5 to 1.6 seconds to process an image of 256x256 pixels and is tested in MatLab.

B. *Motivation for the research*

Considering the existing texture classification methods [2, 3, 4] and technologies, we came to the conclusion that they are not effective for textures having identical structures, they need significant computational time, and in the most cases cannot obtain high recognition accuracy. The existing neural network software for texture recognition is applicable for simulating and testing the methods but is not intended for implementation in standard modules as PLCs widely used for control of different automated technological processes in industry.

The paper structure consists of MSNN and functioning modes description, definition of image processing and texture feature extraction. As next training of the neural network structure is described and the results of the recognition of seven wood textures are represented and discussed. The results are compared to these obtained by similar approaches. The main benefits of the developed system, together with the applied method are discussed.

II. SYSTEM STRUCTURE

The designed system for texture recognition of wooden surfaces with a modular recognition structure of neural networks (NN) is implemented in a PLC for real-time operation as shown in Fig. 1. The system uses only standard devices and interfaces such as a high resolution (Charged Coupled Device) CCD camera, standard personal computer (PC), Siemens PLC model S317 and technological terminal (TT), Siemens Operational Panel (OP) 73 with liquid crystal (LC) display. The Siemens proprietary Multi Point Interface (MPI) - (based on the standard EIA-485) is used for connecting the PC to the PLC. The system functionality is described in the next section.

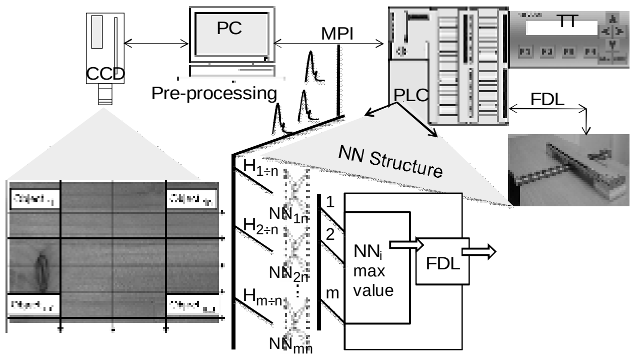


Figure 1. Experimental MSNN with PLC implementation

III. FUNCTIONAL MODES

The system works in two modes – Off line or pre-processing and training mode; On-line or recognition and classification mode.

In “off-line” mode, after image acquisition of a large wooden surface, MxN dimensional grid with M rows and N columns is applied on the image. The histogram of each MxN cell is calculated and defined as a different class parametrical description. The correlation coefficients r_{ij} between each two classes are calculated according to [5]. All classes with r_{ij} greater than 0.75 (i.e., they are high correlated having almost the same histograms) are attributed to one and the same new class. Thus, the cells containing some defects will be separated in a different class because of a small correlation with the dominant surface background - Fig. 3. The class definition is performed with different samples of the same texture some of which with different defect inclusions. Next step in this mode is training of M different neural networks (NNs) with the obtained histograms, corresponding to each defined class. The “off-line” mode operation takes place completely in the PC. After finishing this mode, the trained NNs are downloaded to the PLC in different function blocks.

In “on-line” mode the chosen MxN grid is applied to the current CCD camera acquisition and the histograms are calculated for each cell. Each histogram column (1÷n) of the same raw (M) is used consecutively as input data to the corresponding NNmn in the PLC (Fig. 1). The recognized class results in a maximal value at one of the outputs in the corresponding NN. Additional logic for the assessment of the NN output values and for the finding the maximum one is developed in the PLC using ladder diagram technique. In case of good homogeneous surface, all NNs give the same output with maximum value i.e., one and the same recognized class. If a defect or other inhomogeneous area exists, the corresponding NNmn recognizes one of its input samples as another class. The defected region is defined as a cell that is a cross-point between raw (NN number) and column (number of the current NN input sample) in the MxN grid. A Final Decision Logic (FDL) in the PLC is developed to combine grid cell regions, recognized as one and the same class and to announce them as corresponding homogeneous areas. Finally, the FDL classifies the grid cells with some homogeneity variations and all the rest homogeneous areas. A technological terminal is used to assign the two system modes and display the classification results. FDL results are directly usable for the purpose of connecting to the control logic, actuating the physical outputs of the PLC. Next step is to analyze the feasible combinations between structural ranges for wedge-shaped splicing of wooden balks and to develop logic for any technological process control.

IV. IMAGE PROCESSING

The image preprocessing algorithms are executed on a separate PC after acquiring the image from the camera. The generalized block scheme of these algorithms is presented on Fig. 2. These algorithms are very important since the recognition of the objects depends on their results, as well as on the overall system performance.

The first step after receiving the image from the camera, is to extract the region, which contains only the inspected object by applying edge detection algorithms (and calculating the rotation angle if required). The second step is to segment the image into M rows and N columns. The values N and M are user defined and they depend on the size of the object and the required texture analysis accuracy. After segmenting the image into objects (O), a parallel execution of histogram calculation for each object is performed. The number of the parallel processes is N. Following the parallel calculation of the histograms, each of the results form an input data vector, that is send in serial order ($O_{X,0}$ – $O_{X,N}$) to the neural network for further analyses.

V. NN TRAINING – EXPERIMENTS AND RESULTS

Each NN in the MSNN is of type multi-layer perceptron. Each NN topology was optimized applying the method given in [6]. The method is based on changing the number of hidden layers, number of neurons in each layer and reducing the mean square error (MSE) till all NN outputs have regions, where the output stays near to “1” or “-1” as shown in Fig. 4b. Finally, all NNs were trained with MSE between 0.05 and 0.1 and different

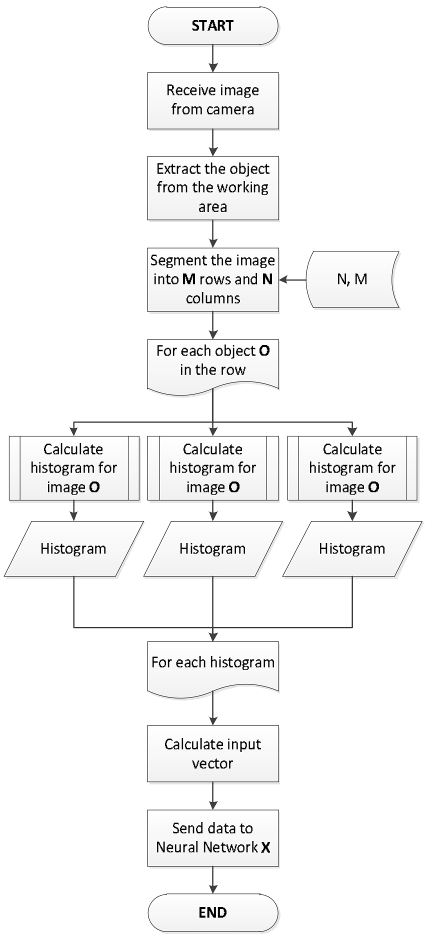
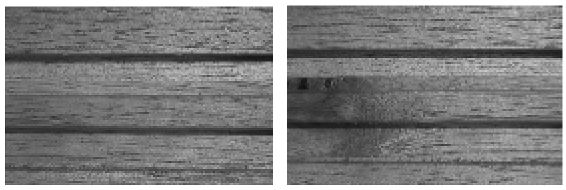
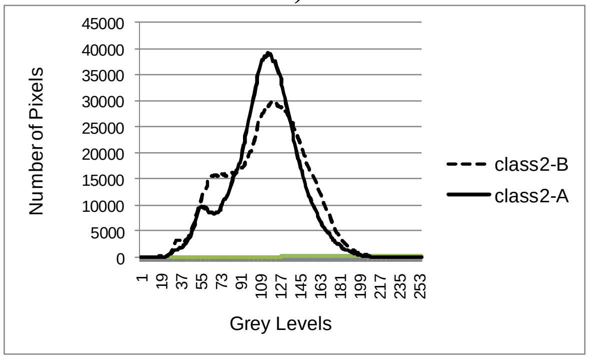

START
Receive image from camera
Extract the object from the working area
Segment the image into **M** rows and **N** columns
N, M
For each object **O** in the row
Calculate histogram for image **O**
Calculate histogram for image **O**
Calculate histogram for image **O**
Histogram
Histogram
Histogram
For each histogram
Calculate input vector
Send data to Neural Network **X**
END

Figure 2. Image preprocessing algorithm



a)

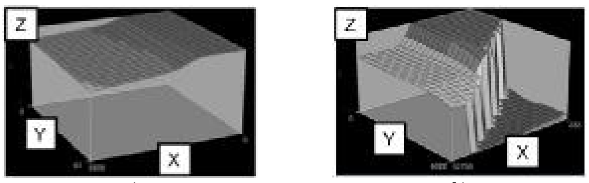


b)

Figure 3. a) Two cells defined as class2-A and class2-B
b) The calculated histograms of class2-A and class2-B

topologies: 42-20-10, 42-50-10, 42-80-10 where 42 (input neurons) is the number of sampled histogram values, 80 is the chosen optimal number of hidden layer neurons, and 10 (output neurons) is the number of recognized wooden surface classes. Tangent hyperbolic was used as NNs transfer function. Each trained NN is downloaded to the PLC in a different Function Block (FB) using NeuroSystem Run-time tool [7]. When calling the FBs, the NN inputs must be provided with the desired values, i.e., the addresses of where the inputs are stored should be stated. After being processed by the FB, the values written to the outputs are read out from the instance Data Block (DB).

The maximal number of different NNs is 256. The classification performance was validated with a histogram dataset different from the training dataset. 1100 training samples and 100 test samples were used. Fig. 5 shows the recognized classes given in Fig. 3, 6 and 7. Each $NN_{1n} \div NN_{5n}$ recognizes (x) the corresponding sample (column in the MxN grid) as a predefined class in the training phase. For example the defected texture given in Fig. 3a will be recognized as class 2-A in the outputs of $NN_{1n} \div NN_{2n}$ for all *n* samples (in our experimental grid n=4) and also for samples 3, 4 of $NN_{3n} \div NN_{5n}$. Samples 1, 2 of $NN_{3n} \div NN_{5n}$ are recognized as class 2-B. The recognition accuracy given in % (Fig. 5) is calculated as the overall number of correct classifications, divided by the number of samples in the testing dataset for each class.



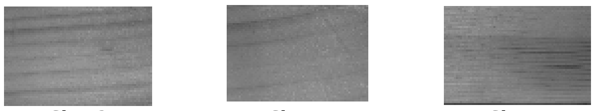
a) *b)*

Figure 4. a) Undesirable change of a NN output Z depending on two inputs X, Y of the corresponding histograms in training phase
b) Desirable change of a NN output Z depending on two inputs X, Y of the corresponding histograms in training phase

class / NN	NN_{1n}	NN_{2n}	NN_{3n}	NN_{4n}	NN_{5n}	recognition accuracy [%]
class1-A	xxxx	xxxx	x xx	xxxx	xxxx	88
class1-B			x			93
class2-A	xxxx	xxxx	xx	xx	xx	98
class2-B			xx	xx	xx	98
class 3	xxxx	xxxx	xxxx	xxxx	xxxx	100
class 4	xxxx	xxxx	xxxx	xxxx	xxxx	100
class 5	xxxx	xxxx	xxxx	xxxx	xxxx	100

Figure 5. Recognition (x) of a corresponding sample (column in the MxN grid) for classes given in Fig. 3,6,7.

Fig. 7a) and b) shows the output values of all NNs (m=1÷5) for columns 1 and 3 respectively when recognizing texture surface in Fig. 1. For column 3 all NNs give the same result, till for column 1, NN_{31} and NN_{41} recognize a different class. The maximal value of the time needed for image acquisition, histogram calculation, and providing the values to the PLC NN Function Block, recognition through the trained NN, and sending the result to the text display OP73 takes 579 ms on the average with a Siemens PLC model S317. The obtained recognition accuracy is between 93% and 100% for real-time performance in the PLC.A complete model for the inspection system is shown on Fig.



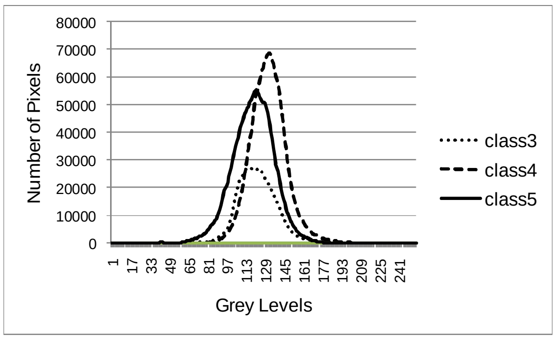
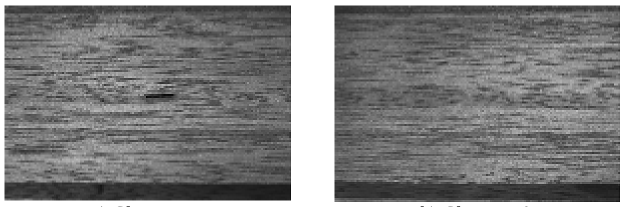


Figure 6. a), b) and c) Three cells defined as classes 3,4,5
d) The calculated histograms of classes 3,4,5

A system prototype has been built in the laboratory "Intelligent Manufacturing Systems" at the Technical University of Sofia, Bulgaria.



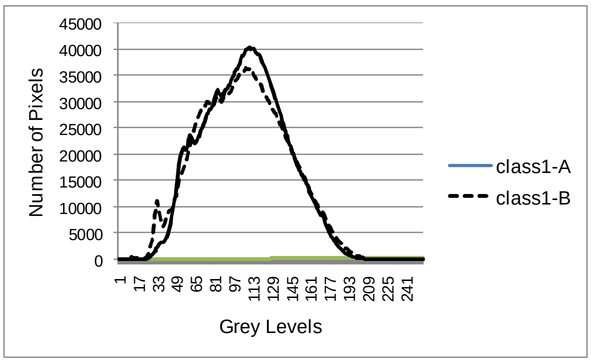
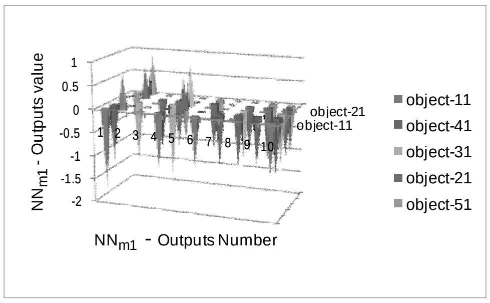


Figure 7. a) and b) Two cells defined as class1-A and class1-B
c) The calculated histograms of classes 1-A and 1-B

The system functioning is quasi-parallel because after the texture acquisition and image segmentation in mxn cells, the histogram calculation goes in parallel, but the histogram downloading as input parametrical vectors to the NNs in the PLC is accomplished as a serial communication.

VI. COMPARISON OF THE RESULTS

Many similar approaches [8, 9, 10] have the common disadvantage that they are not intended for implementation in standard modules as PLCs widely used for control of different automated technological processes in industry. The results of our research can be compared to the neural network approach given in [2]. The proto-type PC-based wood recognition system is capable of classifying 30 different tropical Malaysian woods according to their species based on the macroscopic wood anatomy. Image processing is carried out using newly developed in-house image processing library referred to as "Visual System Development Platform". The textural wood features are extracted using a co-occurrence matrix approach. A multi-layered neural network based on the popular back-propagation algorithm (BPG) is trained to learn the wood samples for the classification purposes. The system can provide wood identification within seconds. The results obtained show a high rate of recognition accuracy. Several MLP models (with different structures) are trained by the authors, using 20 input features (Angular Second Moment, Contrast, Correlation, Inverse Difference Moment, Entropy etc., for five different image rotations), extracted from the texture. In this case, the authors have obtained 1sec overall computation time and 95% recognition success of 20 different tropical wood species. But this system is not intended for implementing the recognition results in an automated system for further process control.



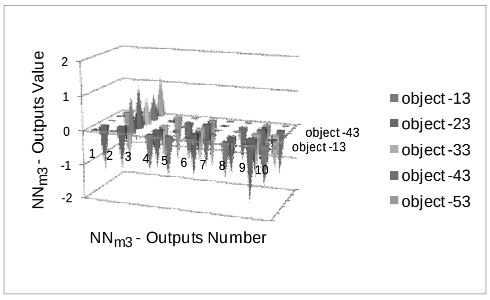


Figure 8. a) Output values of all NNs (m=1÷ 5) for column 1 when recognizing texture surface in Fig. 1
b) Output values of all NNs (m=1÷ 5) for column 3 when recognizing texture surface in Fig. 1

In our case, the overall computation time is 579 ms and the obtained recognition accuracy is between 93% and 100% for real-time performance in a PLC S7-317. The proposed in this paper MSNN system provides that the techniques used is suitable to be implemented for industrial purposes. The development of additional FDL working in "on-line" mode is used for assessment and further logical interpretation of NNs outputs at the recognition stage. FDL results are directly usable for connecting to the control logic, actuating the physical outputs of the PLC.

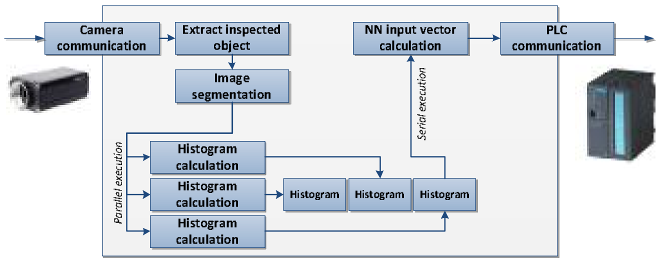


Figure 9. Model of the inspection system

VII. Conclusion

- A MSNN is designed, implemented, and proved for real-time work on standard PLC SIMATIC S7-317 instead on FPGA, because PLCs are widely used as control devices in automated production.
- The parallel work of many different NNs, which number is practically limited to the PLC functionality (256 MLP NNs in our case), provides a good opportunity for using different texture (or objects) parametrical description data as input data for the NNs; Additional FDL is developed for assessment and further logical interpretation of NNs outputs at the recognition stage (in On-line mode);
- MSNN has been tested for classification of wooden textures. It shows high recognition rate and fast performance for the tested samples;
- FDL results are directly usable for connecting to the control logic, actuating the physical outputs of the PLC. The developed MSNN includes standard devices and interfaces and is easily stackable with already introduced different automated control systems in many automated productions at a low price, affordable for small and medium enterprises (SMEs).

References

[1] http://www.weinig.com/ , last accessed on 06.4.2011
[2] Marzuki, K., et al., "Design of an Intelligent Wood Species Recognition System", IJSSST, Vol. 9, No. 3, pp. 9-19, September, 2008.
[3] Grassi, A.P., et al., "Illumination and model-based detection of finishing defects", Reports on Distributed Measurement Systems, Shaker Verlag, pp. 31-35, 2008.
[4] Puente, F. and Dostert, K., "Reports on Industrial Information Technology", Karlsruhe Institute of Technology, Vol 12, pp. 35-54, 2010.
[5] Widrow, B. and Stearns, S., "Adaptive Signal Processing", Prentice-Hall Inc., Englewood Cliffs, N.J. 07632, pp. 36-40, 2004.
[6] Topalova, I. and Tzokev, A., "Automated Classification of Marble Plate Textures with MLP-PLC Implementation, International Conference on Engineering and Meta-Engineering", ICEME 2010, pp. 115-119, 6-9 April, Orlando, USA, 2010.
[7] NeuroSystem V5.0 User Manual, Siemens GmBH, pp. 169-187, 2005.
[8] Iivarinen, J., "Surface defect detection with histogram-based texture features", In SPIE Intelligent Robots and Computer Vision XIX: Algorithms, Techniques, and Active Vision, volume 4197, pp. 140–145, 2000.
[9] Brandtberg, T., "Individual tree-based species classification in high spatial resolution aerial images of forests using fuzzy sets." Fuzzy Sets and Systems 132(3), pp. 371-387, 2002.
[10] Tremeau, A. and Borel, N., "A Region Growing and Merging Algorithm to Color Segmentation", Pattern Recognition, Vol. 30, No.7, pp. 1191-1203, 1997.

Efficient Web-based Monitoring and Control System

Ahmed M. Mohamed
Electrical Engineering Dept.,
Aswan Faculty of Engineering,
Aswan, Egypt
ahmed@engr.uconn.edu

Hosny A. Abbas
Qena Paper Company
Qena, Egypt
hosnyabbas@yahoo.com

***Abstract*— Computer-based supervisory control and data acquisition (SCADA) systems have evolved over the past four decades, from standalone, compartmentalized operations into networked architectures that communicate across large distances. There is an emerging trend comprising SCADA and conventional IT units toward consolidating some overlapping activities. This trend is motivated by cost savings achieved by consolidating disparate platforms, networks, software, and maintenance tools. For reasons of efficiency, maintenance, economics, data acquisition, control platforms have migrated from isolated in-plant networks using proprietary hardware and software to PC-based systems using standard software, network protocols, and the Internet. In this paper, we present a new approach for web based SCADA systems that adapt to the behavior of the target application. In addition, we take into account the real time constraints that imposed by the nature of the problem. We show that our approach is more efficient than other approaches in terms of consuming as little as possible of the available resources (computational power and network bandwidth).**

Keywords – Automation; SCADA; OPC DA; Web Services.

I. Introduction

As the technical capabilities of computers, operating systems, and networks improved, organizational management pushed for increased knowledge of the real-time status of remote plant operations. These capabilities are known collectively as Supervisory Control and Data Acquisition or SCADA. As the name indicates, it is not a full control system, but rather focuses on the supervisory level. SCADA systems are vital components of most nations' critical infrastructures. They control pipelines, water and transportation systems, utilities, refineries, chemical plants, and a wide variety of manufacturing operations. Automation technology produces large amounts of data. This typically represents physical variables such as temperature, current, pressure or other production data such as number of units, error information and so on. The data comes from equally diverse sources: controllers, level measuring devices, weighing machines, scanners and similar. These data end up being crunched or otherwise retained by all sorts of software applications. It may need to be placed straight into visualization software for HMI (Human Machine Interface).

One of the most familiar automation protocols now is OPC protocol. OPC stands for OLE (Object linking and Embedding) for Process Control or Open Process Control [2,9]. It provides a mechanism to provide data from a data source and communicate the data to any client application in a standard way. A vendor can now develop a reusable, highly optimized server to communicate to the data source, and maintain the mechanism to access data from the data source/device efficiently**.** Providing the server with an OPC interface allows any client to access their devices. The utility of OPC has now reached the point where automation without OPC is unthinkable. This interface is supported by almost all SCADA, visualization, and process control systems [3]. OPC delivers connectivity and interoperability benefits to measurement and automation systems in the same way that standard printer drivers deliver connectivity and interoperability benefits to word processing. Both OPC server and OPC clients are COM objects. It makes no difference who developed these objects, when they were developed, or in what programming language they were written. Not only the services of the OPC servers on the same computer are available but also those of all servers which can be accessed over the network. Our objective is to integrate the existing OPC server data access (OPC DA) with the Internet to achieve an efficient web-based SCADA system. In our approach we consider many design parameters such as how frequent the target data change, the consumption of the available resources and the real time constraints that imposed by the nature of the application. This paper is organized as follows; in Section II we give a brief background for some of the solutions that have been developed for this problem. In Section III, we present the motivation behind our research. We introduce our approach in Section IV. In Section V, we present evaluation for our approach.

II. Background

There are many solutions that have been proposed to solve this problem. These solutions can be classified into four categories.

A. DCOM

Most of the previous work used DCOM for LANs communication with OPC DA server for example, Xiaofeng Lee et al. [18] used DCOM communication between an OPC DA client and OPC DA server then they transfer the OPC DA client data to XML format to be able to access these data through Internet with an XML-DA client which communicate to an XML server (Web server) to get data. Truong Chau et al. [16] used DCOM to enable the C# server script to access OPC DA through LAN and because the OPC DA client is a .NET client they used an OPC .NET wrapper to make the transformation from .NET to COM and COM to .NET. Zhang Lieping et al. [19] uses the OPC DA Toolbox which is integrated in MATLAB 7 and above editions, this Toolbox enables MATLAB applications to communicate with OPC DA servers through DCOM then the user can simply and conveniently realize the operation to the OPC objects. Unfortunately, using DCOM through Internet is avoided for many reasons such as, DCOM is windows dependent platform, difficult to configure, has very long and non-configurable timeouts, and cannot be used for Internet communication. That is why DCOM is best suitable for LANs where there are less number of nodes and small delay times.

B. XML

XML is a platform-independent which is an important feature to achieve interoperability between different applications that is running on different platforms. The other advantage of OPC XML-DA is the simple administration as it is based on SOAP and XML. Xiaofeng Lee et al. [18] suggested designing an information integration system which will adopt OPC DA to OPC XML-DA; the design includes three layers structure, data source layer, data transfer layer and client layer. His conclusion was that because of adopting industry standard OPC interface and web service transfer interface, the remote monitoring system based on OPC XML-DA technology makes it convenient to update and expand system. But if we analyze this approach we find that there is an overhead in layer2 because of the COM-XML transformer, also the authors didn't expose to the problem of client data update is there a data polling mechanism that enable the client to get the new data. Similar work used XML and/or OPC XML-DA techniques such as [3, 15, 17]. Due to possible performance limitations, OPC XML-DA is unlikely to be used for real time applications, although it is commonly used as a bridge between the enterprise and control network.

C. Webservice

Webservices as defined by the W3C as "a software system designed to support interoperable machine-to-machine interaction over a network. Nunzio M. Torrisi et al. [8] proposed what they called CyberOPC which is a communication system that anticipates the use of a gateway station called CyberOPC, which will process messages sent to the OPC towards the public network and vice versa. Their proposed communication system is targeted to best effort network with minimum bandwidth reserved for periodic traffic. As they claimed that the necessity to satisfy time-critical and security requirements for remote control has stimulated the study of a new protocol for process control. And to obtain maximum interoperability with existing factory floor technologies, they built their communication project over the OPC technology. This approach doesn't solve the problem of periodic data requests from the remote client to update its old data with new data, and when the remote client makes a request, the CyberOPC will not check if there is a change in these data from the last sent data or not so if the control process has a high frequency data change, the remote client has to increase its periodic data requests which will affect the server efficiency and network bandwidth.

Duo Li et al. [3] suggested an approach to implement the function of real time monitoring which provide periodically updated data to operators, the target data which saved in a database is marked and mapped to an HTML file, say file2, in a web server, the database server automatically refreshes file2 with the latest data whenever the target data in the server is updated. To show target data in an HTML file a Java applet is developed that connects to file2 to get the latest data periodically and display them in the required format, another HTML file ,say file1,in which the applet is included, is developed to establish a basic human-machine interface (HMI) and complete some initiations.

D. AJAX

Actually AJAX (Asynchronous JavaScript and XML) [5] started a new line in web based SCADA systems because it enables us to create more interactive web pages which are our way to replace traditional desktop SCADA application. AJAX can be considered as the future of the Internet because of its ability to simulate desktop applications by the new features it offers like asynchronous client-server calls and partial-page updates. Truong Chau et al. [16] used AJAX in their approach to get the OPC DA data to the client web page with high user interactivity. But because of the problems of the HTTP protocol as a stateless protocol they had to use a simple data polling mechanism by using an AJAX Timer to poll for new OPC DA data set from the OPC DA server through the web server. When using an AJAX style of programming the old, classic programming approach must be given up. There is no form submit any more that posts all the client state to the server and requests for a complete new page description using HTML. Instead of loading several pages until the functionality is done only one page is loaded and will stay in the browser until the end of functionality.

III. MOTIVATION

None of the mentioned approaches considered the feasibility of the realization of their work. If we look closely to [3, 13, 15, 17] we will find out that they use XML and/or OPC XML-DA technologies which have disadvantages such as:

- Not suitable for transferring large data volumes
- XML technology is generally slower than COM
- The interaction parameters are coded using XML, which leads to an overhead.
- An OPC XML-DA Service is stateless.

Also in [12], to allow each node (client or server) to start sending data to the other node independently at any time (bidirectional communication) the authors have to maintain the connection in a permanent open state by making the server continuously sending data (whether there is a data change or not) to the client which affects the performance of the system specially with large number of clients and low network bandwidth, and this may lead to server crash.

In [14], the authors will need to find a way to periodically update file2 which may be a loop or any other way, also the java applet which embedded in the HTML file1 will need to periodically use a Timer to get the new data from file2 and then they will face the problem of synchronization between the applet Timer interval and the process of updating file2 with new data and this will be difficult, also they ignored the heavy network load and the need for large server memory to handle the large number of requests and the heavy load on the server CPU. It is very difficult (nearly impossible) to guarantee a real time behavior with such limitations. The only solution is to spend money to increase the network bandwidth and the server CPU speed and memory capacity (The hardware resources). In our work we try to find alternative, much cheaper solutions to these problems. Also in [8], the designed CyberOPC doesn't solve the problem of periodic data requests from the remote client to update its old data with new data, and when the remote client makes a request, the CyberOPC will not check if there is a change in these data from the last sent data or not so if the control process has a high frequency data change, the remote client has to increase its periodic data requests which will affect the server efficiency and network bandwidth.

Finally, in [16], the authors did not take any attention to the efficiency and performance of their approach, the Timer they used will get the data without any care if there is a significant data changed or not, also how they could specify the Timer interval, there are two cases for that, first, when the interval is small i.e. 1 second (high frequency data polling) by this way the server and network will suffer, second if the interval is large they will loss some data change events that maybe very important and the system will not be considered as a Real-Time monitoring system. So this approach still needs some modifications which enable us to get optimal efficiency and performance, for this reason we choose this approach to be our reference approach. We believe that AJAX can be the way to make web applications compete with desktop applications by the interesting features it offers [5].

IV. The Proposed Approach

The main challenge will be to design and implement a new approach for providing a direct web access to controllers using OPC DA server that is aware of the consumption of the available resources and try to fulfill the real time constraints as well. We have two options to achieve this target, first using DCOM, second using a web server and modern IT technologies. If we used the first option, we will face the DCOM problems (see section II). Therefore, we will use the second option, which is web server and modern IT technologies. However, with this option we are going to face many design and implementation challenges such as, HTTP limitations, responsiveness, real time constraints and the efficiency of the approach. In a classic client/server application, there is a static communication link between client and server but with web applications, this static communication does not exist thus the need for a persistent communication mechanism by which a client can communicate with the server independent of any specific action taken by the user. We want that communication take place even if the user is not clicking or using any of the controls in the user interface.

A. System Overview

Our target system consists of a web server, an OPC server and a Programmable Logic Controller (PLC) units which will be connected in the same plant network, which may be industrial Ethernet network or Fieldbus network (Profibus, MPI... etc), in our case it will be an Ethernet LAN because it is now a standard and familiar protocol in industry. This LAN connected to internet through an ADSL router which has a local (virtual) IP number and an international (Real) IP number. A Client (browser) can connect to the Webservice residing in the web server though internet using HTTP protocol, then the Webservice will connect to the OPC server in the same LAN using DCOM to get or set the new Data as shown in Figure 1. To achieve our goals we will use a modular architecture in the design of our web based SCADA system. As shown in Figure 2. Our system consists of three main modules, which are:

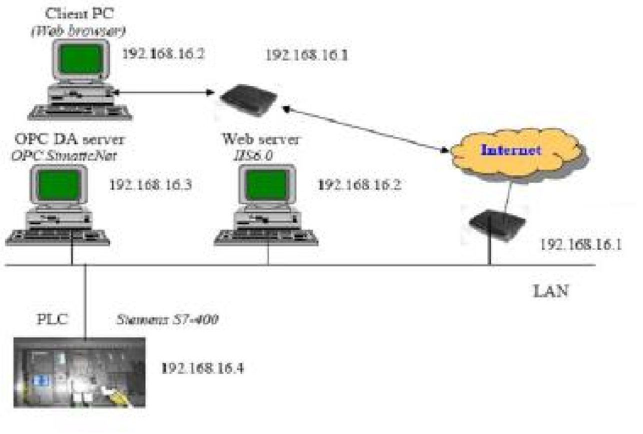


Figure 1: Proposed System Overview

1. The Client, which is a web browser that will be used to run and display the designed SCADA web page and will send the required requests to update the web page with the new data. The requests will be sent asynchronously (in background) by the AJAX engine and the web page will still be responsive. When the request's response comes, the AJAX engine will forward the returned data to the web page then the web

page will send another request and so on. A new request will be sent only if the response to its predecessor request comes.

2. The web server which will wait for the HTTP client's requests and forward these requests to the Webservice which in turn will run the proper web method (WM1 if the request asks for new data and WM2 if the request contains set-points to the OPC DA server). The WM1 will wait for a data change in the OPC DA server cache and then return the new data to the client. WM2 will write the coming client's set-points to the OPC DA server and return to the client without waiting. Also, it includes an OPC DA Wrapper**,** as we mentioned that OPC DA protocol is based on Microsoft COM technology. However, the Webservice, which we use to access this protocol, is implemented in modern .NET technology. So, we will have to find a way to access a legacy technology from a modern technology, the OPC FOUNDATION solved this problem by creating wrappers to enable .NET clients to access COM servers.
3. The OPC DA server, which is the interface to the physical control process (PLC), the OPC DA server has two layers, one to provide the standard COM interface to its clients and another one, which is a vendor specific driver to communicate with the PLC.

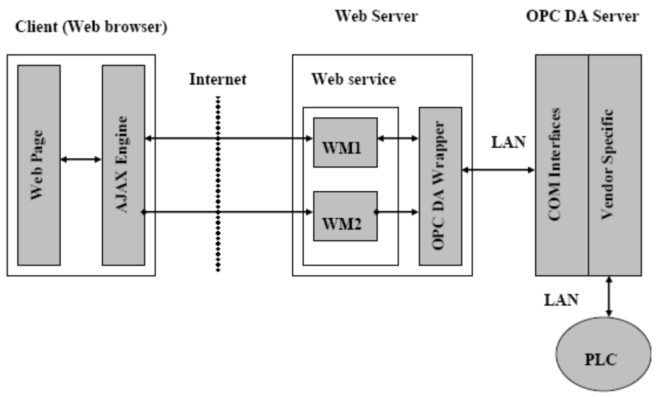


Figure 2: Architecture of the Proposed System

B. *Proposed Approach Design*

From the discussion of previous work and because of the Internet infrastructure, we conclude that data polling is the only solution, which is robust and viable; our main challenge will be implementing a data polling that is effective and wastes as little as possible of the available resources**.** A traditional data poll such as the one used by Truong Chau et al. [16] will query the server, get an answer, and then wait until the next query. The client side has to choose the rate of polling the server for a new data (i.e., a request every 5 seconds) without taking into account the frequency of data change (fast or slow). The waiting period until the next poll is a dead time during which neither the client nor the server can communicate with each other.

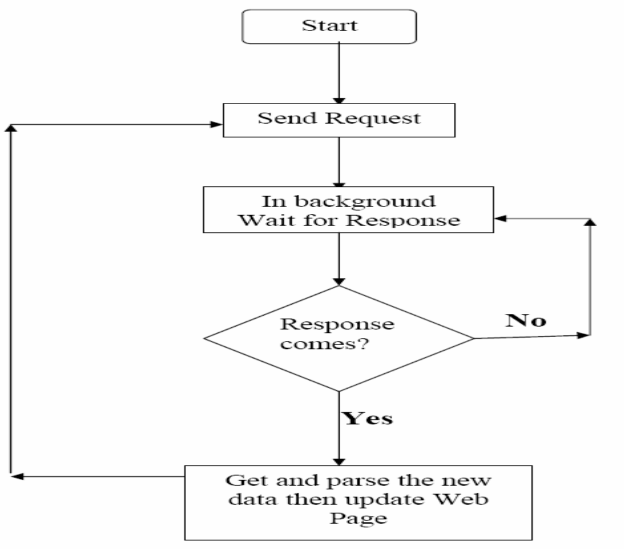

Start
Send Request
In background Wait for Response
Response comes?
No
Yes
Get and parse the new data then update Web Page

Figure 3: The client side

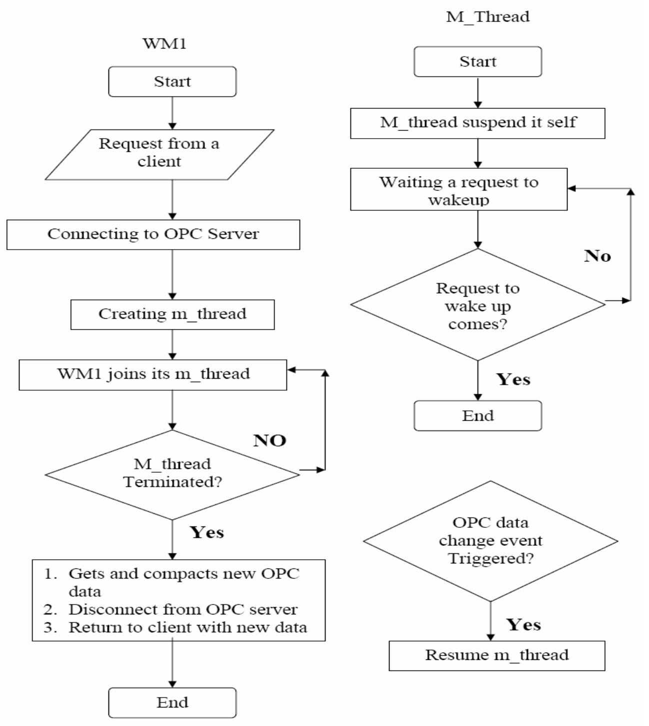


Figure 4: The server side Webservices

In our approach we do not choose a constant data query period instead we adapt to the frequency of data change of the application. This can be achieved by converting the dead time into a wait time created by the web server. The client sends a request and waits for the potential of data change. The server responds to the client only when there is a data change. Otherwise the request is put in a waiting queue at the server. The design flow charts of our approach is shown in Figure 3 (for client side) and Figure 4 (for server side). For complete description of the design and the code of each module, please check [1].

V. EVALUATIONS

Any control process has a number of measurable physical quantities, which are measured by different sensors, and then the raw signals transferred to the PLC to be converted by the A/D converters to digital form to be manipulated by the PLC processing unit. The OPC DA server updates its cache by the new values available in the PLC according to a specified update rate. So a change in the control process quantities leads to a change in the OPC DA server cache. Hence, to simulate a typical control process we will create a software simulation application, which connects to the real OPC DA server, and continuously change its cached data. The simulation application created using VB.NET 2008 and designed to offer many options such as the data change rule (Fixed or Random), the number of data changes (Limited or Unlimited), and the range for Timer interval in case of random data change can be specified, and the changed data range can be specified too. In addition, there is a facility to start and stop data changing at any time to check the behaviors of our proposed approach and other approaches with/without data change. The exact time of each data change can be stored in a database for analysis purposes. In all figures presented in this section we changed the OPC data according to the uniform distribution with parameters (1, 3). Now to show the extent to which we achieved our goals we will compare our approach results to Truong Chau et al. [16] approach results, we will use windows Task Manager to monitor the web server's performance measures and network bandwidth utilization. Applying Truong approach and using simulated OPC data. We choose the timer interval (request rate) for Truong Chau et al. [16] approach to be 1 second as they suggested in their paper. When we run the experiment, we got the CPU usage as shown in Figure 5.

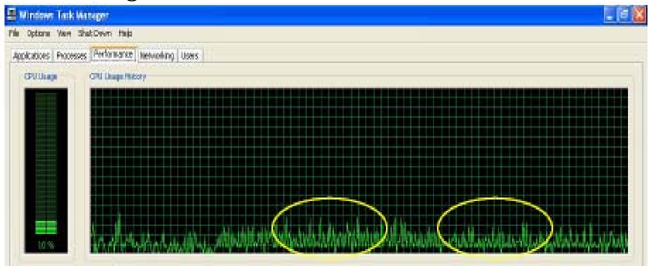

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
CPU Usage
CPU Usage History
10 %

Fig. 5 Truong Approach (CPU load) with Timer interval = 1 sec

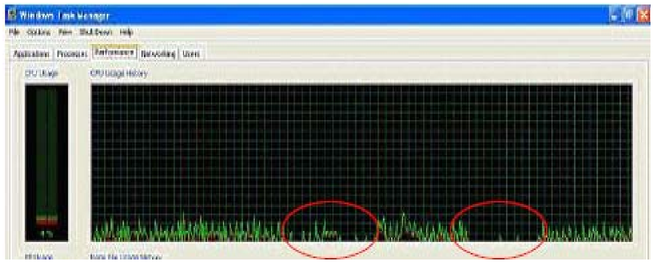
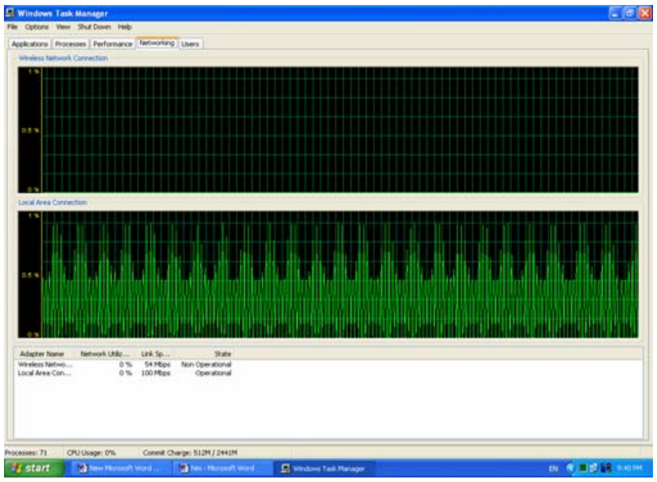

Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
CPU Usage
CPU Usage History
4 %

Figure 6: Our approach test results (CPU Load).

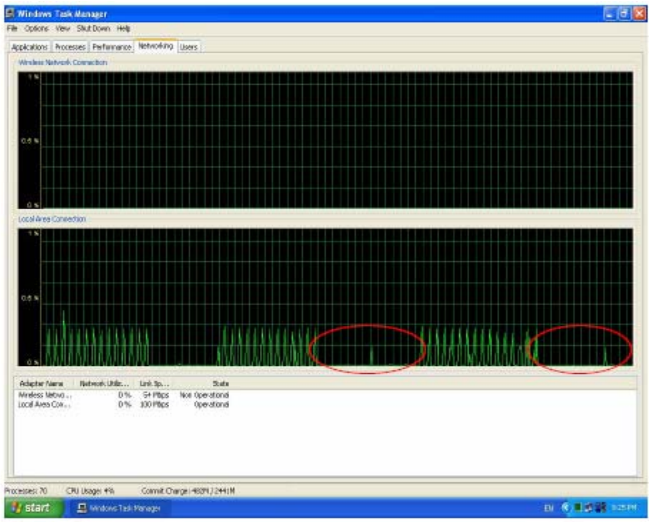
As we see in Figure 5, even if there is no data change (check the circled areas) the server still working approximately with the same CPU usage rate since there is a request coming from the client every one second. However, in our approach the CPU usage is approximately zero when there is no data change as shown in Figure 6 and that because the client does not send a new request until it receives a response to the current request and the server does not send the response until there is a data change. This means that our approach uses the available resources (CPU power and network bandwidth) only when there is a need to do so (significant data change). Thus we use less computational power than previous approaches which proves that our approach is more efficient. Also the data transferred through network with Truong approach is shown in Figure 7, as shown his approach transfers data continuously through network (suppose his timer interval is 1 second) regardless the data changed or not. However, with our approach the data transferred only if there is a data change as indicated by the circled areas in Figure 8.



Adapter Name	Network Utiliz...	Link Sp...	State
Wireless Netwo...	0 %	54 Mbps	Non Operational
Local Area Con...	0 %	100 Mbps	Operational

Processes: 71 CPU Usage: 0% Commit Charge: 512M / 2441M

Figure 7 Truong Approach Network utilization with T =1 sec


Windows Task Manager
File Options View Shut Down Help
Applications Processes Performance Networking Users
Wireless Network Connection
1 %
0.5 %
0 %
Local Area Connection
1 %
0.5 %
0 %
start
Windows Task Manager

Adapter Name	Network Utiliz...	Link Sp...	State
Wireless Netwo...	0 %	54 Mbps	Non Operational
Local Area Con...	0 %	100 Mbps	Operational

Processes: 70 CPU Usage: 4% Commit Charge: 480M / 2441M

Figure 8: Our approach Network utilization

Another experiment we performed using our simulation was to calculate the number of request to the web server for each change in the target application data. We change the rate of data change to be no changes at all, 1 change / 4 seconds, 1 change / 2 seconds, 1 change/second. Applying Truong Chau et al. and our approach we got the following numerical data in Table 1.

TABLE I. NUMBER OF REQUESTS TO THE WEB SERVER

Number of Actual data changes / Minute	**Number of client requests/Minute**	
	Truong Chau. [16] with T=1 Second	***Our approach***
0	**60**	**0**
15	**60**	**15**
30	**60**	**30**
60	**60**	**60**

As we can see, our approach can adapt to the change behavior of the application. The server will send data (use the network) to the client only when there is a data change. While in previous approaches the server keeps sending data to the client even if there is not a data change. For a complete set of experiments, please check [1].

CONCLUSION

As IT technologies progress, web applications will replace desktop applications in most of computer applications fields especially industrial applications like SCADA systems which are now very important in process control. AJAX finally enables us to make an efficient web based SCADA systems because of the features it offers like asynchronous client-server communication and partial page update. In addition, Webservices can be used as server side scripting. Webservices provide a language-neutral, environment-neutral programming model that accelerates application integration inside and outside the enterprise. Application integration through Webservices yields flexible loosely coupled business systems. Using AJAX in a proper way can give us a very good performance in the server and network. In this paper we used AJAX to implement a persistent communication between client and server. In addition, the threading mechanism used in the Webservice was the key concept to achieve this persistent web communication. Because of the Internet infrastructure, we concluded that data polling is the only viable solution to get new data. We showed how our approach for web based monitoring and control system adapts to the behavior of the target application by responding to the client only when there is a data change (the frequency of requests matches the frequency of the application data change). Also, that our approach is more efficient than previous approaches. In addition, we take into account the real time constraints that imposed by the nature of the problem. By using as little as possible of the available resources (computational power + network bandwidth), we improve the responsiveness of the application.

References

[1] Abbas, Hosny and Mohamed, Ahmed "Efficient Web Based SCADA System" Master Thesis, http:// www. engr. uconn. edu /~ahmed/Thesis_Hosny.pdf, Mar. 2011.

[2] Byres research, OPC security WP#1, July 17, 2010.

[3] Duo Li and Yoshoizumi Serizawa, "*Concept Design for a Web-based SCADA system*", Proceedings of Transmission and Distribution Conference and Exhibition 2002: Asia Pacific, IEEE/PES, Vol. 1, pp. 32 – 36, 2002, JAPAN.

[4] http://www.controlglobal.com/articles/2004/229.html, Aug. 18, 2010.

[5] http://www.helium.com/items/580436-advantages-and-disadvantages-of-ajax, Sep 6, 2010.

[6] http://www.w3.org/TR/WD-script-970314, Mar 14, 2010.

[7] Mike Clayton, Philippe Gras, and Juan Oses "A SCADA-WEB INTERCONNECTION WITH TCP IN JAVA", URL: *http://ess.web.cern.ch/ESS/GIFProject/PVSSJava/pvssweb.0.8.pdf*, Nov, 20, 2010

[8] Nunzio Torrisi and João Oliveira," Remote control of CNC machines using the CyberOPC communication system over public networks", The International Journal of Advanced Manufacturing Technology, Vol. 39, pp. 570-577, 2007

[9] OPC Foundation, "OPC DA 3.0 Specification [DB/OL]", Mar.4, 2010

[10] OPC Foundation, "*OPC XML-DA Specification Version 1.0*", Released July 12, 2003.

[11] Shamdutt Kamble, Venkateswara Rao, and Shailendra Jain, "*OPC Connectivity to Remote Monitoring & Control*", White paper, Wipro Technologies company, www.wipro.com.

[12] Shekhar Kelapure, Sastry Akella, and Gopala Rao, "*Application of Web Services In SCADA Systems*", The International Journal of Emerging Electric Power Systems Vol. 6, No 1, pp. 1-15, 2006.

[13] Shaung-Hua. Yang and Lili Yang, "*Guidance on Design of Internet-based Process Control Systems*", ACTA AUTOMATICA SINICA, Vol. 31 NO.1, pp. 56-63, 2005.

[14] Thomas Bauer, Roland Heymann, and Heinz Christoph, "System and method for transmitting OPC data via Internet using an asynchronous data connection", US patent number: US 7,302,485 B2, 2007.

[15] Thomas Dreyer, David Leal, Andrea Schröder, and Michael Schwan, " *ScadaOnWeb – Web Based Supervisory Control and Data Acquisition*", Proceedings of 2nd International Semantic Web Conference, pp. 788-801, 2003, FL,USA.

[16] Truong Chau and Nguyen Khai, "*WEB-BASED DATA MONITORING AND SUPERVISORY CONTROL*", Proceedings of the International Symposium on Electrical & Electronics Engineering. pp. 7-13, 2007, Vietnam.

[17] Vu Van Tan, Dae-Seung Yoo, and Myeong-Jae Yi, "*Design and Implementation of Web Service by Using OPC XML-DA and OPC Complex Data for Automation and Control Systems*", Proceedings of the Sixth IEEE International Conference on Computer and Information Technology pp.263-265, 2006, Korea.

[18] Xiaofeng Lee and Jianfeng Hu,"*Design and Research of Remote Monitoring System based on OPC XML-DA*", Proceedings of 2008 International Pre-Olympic Congress on computer science, V(1), pp. 147-151, 2008, China

[19] Zhang Lieping, Zeng Aiqun, and Zhang Yunsheng, "*On Remote Real-time Communication between MATLAB and PLC Based on OPC Technology*" Proceedings of the 26th Chinese Control Conference, pp. 545-548, 2007, China

A Distributed Multi-Agent Control Model for Railway Transportation System

Ali Pouyan
School of Computer and IT Engineering
Shahrood University of Technology.
Shahrood, Iran.
apouyan@shahroodut.ac.ir

Momeneh Taban
School of Computer and IT Engineering
Shahrood University of Technology.
Shahrood, Iran.
m.taban@comp.tus.ac.ir

Sadegh Ekrami
School of Computer and IT Engineering
Shahrood University of Technology.
Shahrood, Iran.
s.ekrami@comp.tus.ac.ir

***Abstract* –Management and control of railway system is a serious and complex task. A railway system is consisted of network of tracks, stations, safety tools, and trains. In order to cope with increasing train frequencies a method for organizing an appropriate schedule in distributed parties is needed. However this paper concerns the application of distributed multi-agent for presenting an appropriate model consisted of interacted distributed layers for railway transportation system. The proposed layered-based model reduces the workload of the system, which in turn leads to a real-time approach. In this method the system's efficiency increases in comparison with the centralized approach.**

Keywords–distributed; mobile agent; distributed systems; layered-base model.

I. INTROUDUCTION

During the last decade human life has changed by automation revolution. Railway system is one of the systems whose development depends inevitably on automation. Increasing use of railway transportation systems has resulted in a huge complexity system which can not be managed and controlled by human by itself. Various versions of Automatic Train Control (ATC), Automatic Train Operation (ATO), and Automatic Train Protection (ATP) are the paradigms of automation being used in current railway industries [1]; in order to have a complete interoperable railway system, more rigorous research is needed. Despite different technologies being used to implement automation in railway system, for large complex systems, distributed applications are more feasible. Moreover, because of its inherent characteristics, agent technology is capable to design distributed systems [2].

Distributed systems can be implemented using multi-agent technology because it is able to decompose the system into a variety of agent systems that can interact with each other. Accurate interoperability helps these distributed multi-agent systems to reach the main goal of the whole system. Several approaches have been proposed in controlling system with the help of agent technology [3, 4].

This paper presents a multi-agent-based model for controlling and management of railway system. This model is compartmentalized into four layers including central office, local offices, subnets and blocks. All systems applications rely on the communication between these layers. The proposed model helps to reduce system's workload and enhances communication efficiency which leads to a real-time approach. Mobile Agent technology is used to carry on the interaction between layers.

The rest of this paper is organized as follows. Section 2 presents a brief background of the concepts previously being used. Section 3, presents a general distributed model for railway system. Section 4, surveys the system from an internal point of view and determines 4 scenarios which show the real-time functionality of Mobile Agents and Agent Systems. Finally, section 5 concludes the paper with a summary and a discussion of possible future work.

II. BASIC CONCEPTS

A *distributed system* includes a set of autonomous sub-systems linked to each other. Sub-systems are able to coordinate their activities and share the system's resources, information, and etc.

An *agent* is an autonomous computer program accomplishing the assignments on behalf of an agent system.

A *multi-agent system* is a system that consists of multiple interacting intelligent agents. Multi-agent technology usually is used for modeling distributed system in order to solve the problems that are difficult for an individual agent or monolithic system to solve. This kind of system usually uses both mobile and stationary agent to perform its tasks. Different from stationary agent, mobile agent is not bound to the system it begins the work from [5].

A *distributed multi agent system* is a multi-agent system in which individual agents act on different parts of the system.

III. SYSTEM GENERAL MODEL

Most of current traffic management systems are incumbent through only a central control unit (includes central scheduling system, centralized database architecture, operators, etc.) which can cover only a little scope accurately, and only a limited range of potential delays can be performed. The main advantage of the modeled approach is a decentralized planning that relies on distributed databases, distributed processors, distributed rescheduling systems, etc. A general model of the system is shown in Fig. 1.

As it can be seen, this design is consisting of different components whose some tasks are defined as follow:

AEE: Agent Execution Environment is a place for analyzing the operational data of both stationary and mobile agents, managing them, and controlling creation, suspension and immigration of each agent due to its operational essence.

Finder: this component routes and authenticates the messages between layers. It probes for the agent data (origin, destination, etc.) and identifies the agent owner; then transmits it to its destination.

Camera and sensor: The installed detectors wrap the current state of the environment into mobile agents and transmit them to their agencies where they will be unwrapped in order to be analyzed.

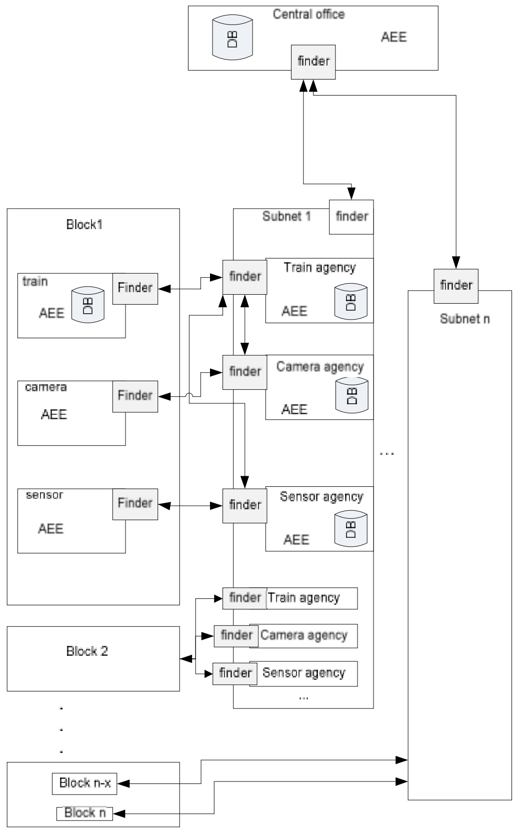


Figure 1. Presents a general model of railway system.

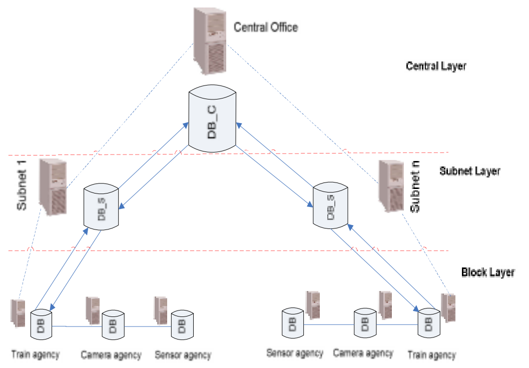


Figure 2. The schema of databases and processors.

Database (DB): Because of distributed modeling, decentralized DB method is used to balance the workload of immense amount of data and transportation task. It helps the system by reducing network traffic, having better availability and better response time. In this method each part controls its own data autonomy. As the Fig. 2 represents it clearly, our system is three layered in terms of database structure. The Central's DB at Central Office is supposed to be in charge of controlling whole databases by collecting integrated data and replicating them. In previous section, we introduced the main parts of our system; we will give a brief description of internal processes in the next section.

Block: Having the best control in an optimized way over rail roads, they are decomposed to some virtual blocks, in which different kinds of detectors (cameras, sensors, etc.) are located. The trains passing the predefined blocks are under block control. Once the trains enter the block, they are identified by that. Each train has a predefined scheduling and it is connected to its agencies in each pulses of the time to send its situation's parameters (speed, location, etc.) through mobile agents.

The whole system consists of blocks, subnets and central office, respectively. The second layer called "subnet" is a group of tripartite agencies. These three agencies (train agency, camera and sensor agency) are in charge of controlling one block. The head agency in each block is the train agency through which scheduling and rescheduling are served. After analyzing the data of camera agency and sensor agency, the output will be transmitted to the train agency.

If each train (for any reason, as mentioned in the following part), causes delay in pre-defined motion schedule, it can not be compensated only by itself; and the adjacent trains (forward, backward) also take part to fix the interval balance. This plan causes the delayed train to have more opportunities to compensate its delay so that the system would not face with dangerous condition.

IV. THE MODEL INFRASTRUCTURES

Trains are considered as equal interval chains along specified routes. If a train encounters an unusual condition and causes delay, the interval between the trains is reduced and it is a critical situation that must be avoided. In fact, upon such a situation, the compensating responsibility will be shared among the trains to balance influences on delayed train.

There are two points of view about the system, one is the system external view that is discussed in the last part, and the other one is the internal view of the system, which consists of all internal processes performed in a system. To perform internal design of a system by using agents, it is required to apply not only agents and their execution environments but also some special functions and mechanisms. This section defines some fundamental and basic processes of agents in connection with agencies trough the pseudo codes which have been showed in the Fig. 3. One of vital points in traffic management field is scheduling system which must be able to deal with problem specification, neglecting the type of technology that is used. Relating to the first point, the second essential requirement is finding an optimal solution depending on problems' type. The following part presents some probable scenarios in which the scheduling system at the train agency, subnet and Central Office are responsible for planning train traffic and finding optimized solutions. Fig. 3 describes brief functional tasks by agents and agent systems.

Trains, sensors and cameras scan environmental state and continuously send a list of data to their agencies within equal time intervals. Analysis information at the train agency takes place when information from trains, sensor agency and camera agency is received. If some potential delay is identified by train agency, different scenarios of system performance are put forward to system delayed state.

There are different critical scenarios which may happen through the train's move and affect secure movement curve (the train’s motion curve that Central Control System sends it to the train, so it can move by using this plan).

To avoid difficulty and complexity of data transmission, the control system is divided into 3 basic layers; the block layer which has the most effective role is Omni-present to deal with the trains. Subnet layer is responsible for interacting with the blocks’ layer in its authority and let them to solve their conflicts whenever the adjacent blocks need to be rescheduled; And Central Layer is designed to control not only the conflicts between subnet layers but also the issues of whole system.

```
Elem= list (train ID, train speed, train velocity, location(x,y), direction, time) // Elem is necessary information of the train
Agent1→ call ('insert', array(Elem)); // It can be sent by the train
Base=Agent2→ call ('retrieve', schedule); // It can be generated by the train agency
Function delay (Train (Elem), Base);
If (delay_result==0) {create Agent (type1) /*Agent type 1 is defined in order to inform the train for continuing its schedule/*} else {
Function curve_Train_Agency (delay_result, predefined compensating algorithms); /* Checks Block Situation, if there is a reschedule in the block calculates a new reschedule, if the reschedule plan is conflicting by another block the reschedule plan will sending its data in an agent form to the higher layer (Subnet) */
If (adjacent block conflict) {call curve_Subnet()} else{ create Agent (type2) /* Agent type 2 is defined in order to change the train's schedule/* } }
Function curve_Subnet (delay_result, predefined compensating algorithms, adjacent block situation); //calculate a reschedule for inter-block schedule.
```

Figure 3. Pseudo code of a conventional relationship between train, train agency and subnet.

To make it more clear, different scenarios are used to explain the model. Blocks are responsible to get the necessary data from trains and verify their characteristics such as ID, location, speed, etc. and compare them with the scheduled plan. The adjacent trains will keep moving even if a train delays for a couple of time intervals. However, in such cases, the plan will be rescheduled immediately by block processors and will be sent to the appropriate trains to compensate the delay. The Mobile Agent role is effectively seen in these communications. It transmits the data which have been reported by different parts such as sensors, cameras, trains. Moreover, agents interacts with each other in an environment called Block Layer in order to verify the errors caused by trains, sensors, etc.

There is a more complicated situation in which the trains in a block make a critical scenario leading to confliction with the adjacent block. In order to reduce the processing work in the block layers, the issue will be referred to the upper layer named subnet layer. The subnet has access to blocks’ databases and reschedules the system in the affected blocks and sends motion curves to the trains.

Consequently, the model will not cause any conflict with any parts, and the delayed time will be distributed among appropriate trains. The system is guaranteed to be encountered with an accident, because if so, the whole motion curve is going to be changed. In cases that problems remain unsolved and get to a critical point, the related blocks will stop the trains. In this paper we do not talk about the functions and details of curve algorithms or any compensating strategy. Our focus is mostly on communication phase, agents’ role, distributed processing, and distributed tasks.

As it is mentioned in the pseudo code, two important functions will be invoked hierarchically. If it is needed to refer to the upper layer and the agents are executed in order to find an accurate schedule. In such a case the upper processors which consist of some lower processors would not be in connection with any ordinary messages and data from the train, so these processors may run in bottlenecks. Hence, the system can save more executive power. The following scenarios show how a schedule plan generates.

1) List of data which are sent by the train will be inserted into the train agency's DB. At the same time the actual schedule is invoked and is compared with the data list. If there is no difference between them an agent will be sent to the train indicating that the previous schedule should be followed.

2) Sometimes the trains available in delayed block can compensate the recognized delay through block rescheduling which is performed by train agency. And the new motion curves are sent to the trains.

3) A critical point is when the adjacent blocks cause delays, or the delayed block could not compensate the delay by rescheduling its block. In these cases, the subnet takes control of the delayed blocks. The subnet shall identify the number of operational risks, such as traffic conflicts, potential delays and bottlenecks, and then organize to avoid them by changing current time tables. Until now the subnet has performed a local planning process for its blocks.

4) As it is mentioned in Fig. 4, another probable situation is when two adjacent blocks from two adjacent subnets cause delay and the compensation process requires the subnets' coordination. In some conditions, probable problems in subnet coordination and subnet schedule can be solved via central office.

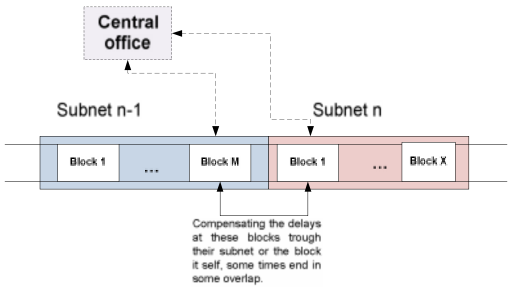

Central office
Subnet n-1
Subnet n
Block 1
...
Block M
Block 1
...
Block X
Compensating the delays at these blocks trough their subnet or the block it self, some times end in some overlap.

Figure 4. Role of Central Office in Controlling Subnets.

V. CONCLUSION AND FUTURE WORK

In summary, this paper presents a four-layer distributed designing to model the general behavior of railway transportation system which consists of different interacting parties. Railway is decomposed to some virtual blocks. We used detectors to control each block accurately and to avoid incidents. To model this distributed system, we used multi-agent system enjoying both mobile agents and stationary agents to reduce network traffic and enhance communication efficiency between distributed agent systems.

Future work is needed to classify and define current agents and agent systems in more details, for instance their exact framework and functions. A more detailed implementation model for future development is also required. Because, some of the aspects, such as authentication, security issue, agent system structures, etc. will become as important as our layered-based model. Finally, simulation fundamentals shall be extended via prospect experimentations.

REFERENCES

[1] K. Takeshi and H. Nakamura, "Automatic Train Protection System Depending on Data Oriented Control Method". Railroad Conference, 1993, pp. 51-57, DOI: 10.1109/RRCON.1993.292963.

[2] D.J. Mendes and M. De Assis Silva, "Mobile Agents in Autonomous Decentralized Systems". The Fourth International Symposium on Integration of Heterogeneous Systems.1999, pp. 258 – 260, DOI:10.1109/ISADS.1999.838444.

[3] B. Chena, H. H. Chengb, and J. Palenc "Integrating Mobile Agent Technology with Multi-Agent Systems for Distributed Traffic Detection and Management Systems". 2009, pp. 1-10.

[4] T.K. Ho, L. Ferreira, and K.H. Law, "Agent Applications in Rail Transportation". Proc of International Conference on Intelligent Agents Web Technologies and Internet Commerce, pp. 251-260, 2004, Vienna.

[5] A. Kavicka, V. Klima and N. Adamko, "Simulation of Transportation Logistic System Utilising Agent-Based Architecture". International journal of simulation modeling 6(2007) 1, pp. 13-24, DOI:10.2507/IJSIMM06(1)2.075.

[6] Z. Linda, B. Naadjib, and E. Aouaouche, "An Architectural Model for a Mobile Agents System Interoperability". 2007, pp. 555-566, DOI: 10.1007/978-1-4020-6270-4-46.

[7] L.M. Gambardella, A.E. Rizzoli, and P. Funk, **"**Agent-Based Planning and Simulation of Combined Rail/Road Transport". May 2002, Vol. 78, No. 5, 293303(2002). pp. 293-303, DOI: 10.1177/0037549702078005551.

[8] R. Leszczyna, "Anonymity Architecture for Mobile Agent Systems". Lecture Notes in Artificial Intelligence; Vol. 4659 Proceedings of the 3rd International Conference on Industrial Applications of Holonic and Multi-Agent Systems: Holonic and Multi-Agent Systems for Manufacturing. pp. 93 – 103, 2007.

[9] M. Miyoshi, M. Seki, Y. Fujiwara, Y. Sobu, and K. Toshiba Corp, "Development in Mass Transmit System", 1998, in Proceedings of the International Conference on Developments in

Mass Transit Systems (Conf. publ. No.453), pp. 196 – 201, ISBN: 0-85296-703-9.

Flexible Adaptation Loop for Component-based SOA Applications

Cristian Ruz, Françoise Baude, Bastien Sauvan
INRIA Sophia Antipolis Méditerranée
CNRS, I3S, Université de Nice Sophia Antipolis
France
{*cruz,fbaude,bsauvan*}*@inria.fr*

***Abstract*—The Service Oriented Architecture (SOA) model fosters dynamic interactions of heteregeneous and loosely-coupled service providers and consumers. Specifications like the Service Component Architecture (SCA) have been used to tackle the complexity of developing such dynamic applications; however, concerns like runtime management and adaptation are often left as platform specific matters. At the same time, runtime QoS requirements stated in Service Level Agreements (SLA) may also evolve at runtime, and not only the application needs to adapt to them, but also the monitoring and management tasks. This work presents a component based framework that provides flexible monitoring and management tasks and allows to introduce adaptivity to component-based SOA applications. The framework implements each phase of the autonomic control loop as a separate component, and allows multiple implementations on each phase, giving enough runtime flexibility to support evolving non functional requirements on the application. We present an illustrative scenario that is dynamically augmented with components to tackle non-functional concerns and support adaptation as it is needed. We use an SCA compliant platform that allows distribution and architectural reconfiguration of components.**

***Keywords*-*Monitoring*; *Management*; *SLA Monitoring*; *Reconfiguration*; *Component-based Software Engineering*.**

I. Introduction

According to the principles of Service Oriented Architecture (SOA), applications built using this model comprise loosely-coupled services that may come from different heterogeneous providers. At the same time, a provided service may be composed of, and consume other services, in a situation where service providers are also consumers. Moreover, SOA principles like abstraction, loosely coupling and reusability foster dynamicity, and applications should be able to dynamically replace a service in a composition, or adapt the composition to meet certain imposed requirements.

Requirements over service based applications usually include metrics about Quality of Service (QoS) like availability, latency, response time, price, energy consumption, and others, and are expressed as Service Level Objectives (SLO) terms in a contract between the service consumer and the provider, called Service Level Agreement (SLA). However, SLAs are also subject to evolution due to different providers, environmental changes, failures, unavailabilities, or other situations that cannot be foreseen at design time. The complexity of managing changes under such dynamic requirements is a major task that pushes the need for flexible and self-adaptable approaches for service composition. Self-adaptability requires monitoring and management features that are transversal to most (all) of the involved heterogeneous services, and may need to be implemented in different ways for each one.

Several approaches have been proposed for tackling the complexity, dynamicity, heterogeneity and loosely-coupling of SOA-based compositions. Notably, the Service Component Architecture (SCA) is a technologically agnostic specification that brings features from Component-Based Software Engineering (CBSE) like abstraction and composability to ease the construction of complex SOA applications. Non-functional concerns can be attached using the SCA Policy Framework. However, concrete monitoring and management tasks are usually left out of the specifications and must be handled by SCA platform implementations, mainly because SCA is design-time and not runtime focused.

Our thesis is that a component-based approach can ease the implementation of flexible adaptations in component-based service-oriented applications. Our proposed solution implements the different phases of the widely used MAPE (Monitor, Analyze, Plan, and Execute) autonomic control loop as separate components that can interact and support multiple sets of monitoring sources, conditions, strategies and distributed actions.

The rest of the paper is organized as follows. Section II presents an example situation that motivates our work and provides a general overview of our contribution. Section III describes the design of our framework from a technologically independent point of view. Section IV presents our implementation over a concrete middleware. Section V describes related work and differentiations with our solution. Finally, Section VI concludes the paper.

II. Motivating example and Overview of Our Contribution

Consider a tourism office who has composed a smart service to assist visitors who request information from the city and provides suggestions of activities and touristic planning. The application uses a local database of touristic events and a set of attraction providers who sell tickets to parks, tours, etc. A weather service guides the proposition of activities, and a mapping service creates a map with directions. A payment service may be needed in some cases. Once all information is gathered, a local engine composes a PDF document and optionally prints it. The composed design of the application is show in Figure 1.

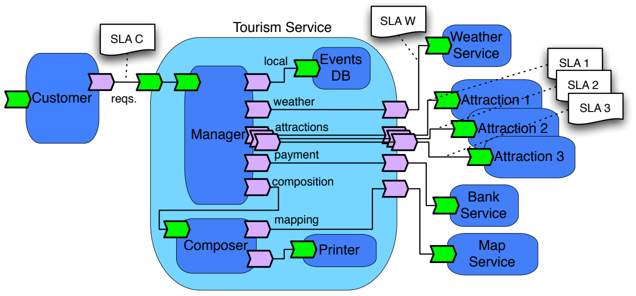


Figure 1. Scenario. SCA description of the application for tourism planning.

Such a composition involves some contracts for service provisioning established in SLAs. For example, the Tourism Service should propose a plan within 30 sec.; the Weather Service charges a fee for each forecast; the Mapping Service is free but has no guarantees on response time or availability; the Banking Service ensures 99% of availability.

The runtime compliance to the SLAs may determine later decisions of the service. For instance, some of the Attraction Services may take too much time to deliver an answer, or the Mapping Service may not be available or have a poor performance at some moment. Situations like those may require that a runtime decision be taken to avoid violating the SLA with the customer. For instance, taking actions like the removal or replacement of a slow provider; or changing some parameter on the composer to ignore the map while composing the document. To be able to make such kind of decisions, a precise and efficient runtime monitoring and SLA compliance system is required.

The monitoring requirements may be different for each service; for example, in the case of the printer it is important to measure the amount of paper or ink; in the case of the composer it is important to know the time it takes to create a document; some of the external services may provide their own monitoring metrics and, as they are not locally hosted and only accesible through a predefined API, it may not be possible to add specific monitoring on their side. This situation imposes a requirement for supporting heterogeneous services and adaptable monitoring

As there are several external providers involved, the conditions expected from each one of them may change, and so the monitoring requirements over them. The Weather Service may decide to modify their charging plans; or some attractions may offer temporary promotions, which may influence the strategy to select them.

The composition of the application may also change. The composer service may decide to provide an alternative bluetooth service to transmit the composed document to a smartphone, less costly than the printer service. In that case, a new component must be added to the composition, and the monitoring and management infrastructure must be changed accordingly.

A. Concerns

As it can be seen from the example, concerns about SLA and QoS can be manifold. A monitoring system may be interested in indicators like performance, energy consumption, price, robustness, security, availability, etc., and the set of required values may be different for each monitored service. Plus, not only the values of these indicators change at runtime, but also the set of required indicators, as the monitoring requirements can also evolve. Moreover, due to the heterogeneous nature of the providers, some of the services may require specific protocols to retrieve monitoring values or to perform modifications on them.

In general, the evolution of the SLA and the required indicators can not be foreseen at design time, and it is not feasible to prepare a system where all possible monitorable conditions are ready to be monitored. Instead, it is desirable to have a flexible system where only the required set of monitoring metrics are inserted and the required conditions checked, but as the application evolves, new metrics and SLA conditions may be added and others removed minimizing the intrusion of the monitoring system in the application.

B. Contribution

We argue that a component-based approach can tackle the dynamic monitoring and management requirements of a composed service application while also providing self-adaptivity. We propose a component-based framework to add flexible monitoring and management concerns to a running component-based application.

In this proposition we separate the concerns involved in a classical autonomic control loop (MAPE) and implement those concerns as separate components. These component are attached to each managed service, in order to provide a custom and composable monitoring and management framework. The framework allows to build distributed monitoring and management architectures that are associated to the actual functional components in an integrated way. Our framework leverages the monitoring and management features of each service to provide a common ground in which monitoring, SLA checking/analysis, decisions, and actions can be carried on by different components, and they can be added or replaced separately.

We believe that the dynamic inclusion and removal of monitoring and management concerns allows (1) to add only the needed monitoring operations, minimizing the overhead, and (2) to better adapt to evolving monitoring needs, without enforcing a redeployment and redesign of the application, and increasing separation of concerns.

III. Design of the Component-Based Solution

Our solution relies on the separation of the phases of the classical MAPE autonomic control loop. Namely, we envision separate components for monitoring, analysis, planning, and execution of actions. These components are attached to each managed service.

The general structure of our design is shown for an individual service C in Figure 2. Service C is extended with one component for each phase of the MAPE loop and converted into a *Managed Service* C (dashed lines). The small interfaces are aggregated by our framework to allow interaction with other managed components.

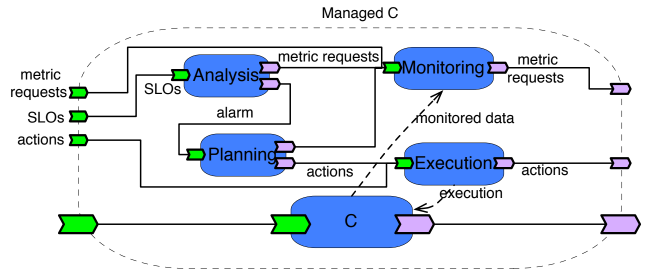


Figure 2. SCA component C with all its attached monitoring and management components

The *Monitoring* component collects monitoring data from service C using the specific means that C may provide. From the collected monitoring data, the *Monitoring* component provides a set of *metrics* through an interface. The *Analysis* component provides an interface for receiving and storing SLOs and checks them at runtime using the metrics obtained from the *Monitoring* component. Whenever an SLO is not fulfilled, the *Analysis* component sends an alarm to the *Planning* component, which uses a strategy to create an adaptation plan, as a sequence of actions. The actions are executed by the *Execution* component, which includes the specific means to make them effective over service C, completing the loop.

Although simple, this component view of the autonomic control loop has several advantages. First, by separating the control loop from the component implementation, we obtain a clear separation of concerns between functional content and non-functional activities. Second, the component-based approach allows to have separate implementations for each phase of the loop; as each phase may require complex tasks, we abstract from their implementation, that may be specific for each service, and allow them to interact only through predefined interfaces, so that each phase may be implemented by different experts. Third, as each phase can be implemented independently, we allow to compose each phase to have possibly multiple components, for example, multiple sensors, condition evaluators, planning strategies, and connections to concrete effectors as required, so that the implementation of each phase can be adapted at runtime.

The framework allows to add and remove at runtime different components of the loop, which means that, for example, a service that does not need monitoring information extracted, does not need to have a Monitoring component and may only have an Execution component to modify some parameter of the service. Later, if needed, it is possible to add other components of the framework to this service.

In the following, we describe the components considered in the monitoring and management framework, their function and some design decisions that have been taken into account.

A. Monitoring

The *Monitoring* task consists in collecting information from a service, and computing a set of indicators or *metrics* from it. The *Monitoring* component includes sensors specific for a service or, alternatively, supports the communication with sensors provided by the target service according to a particular protocol. This way, the *Monitoring* component is effectively attached to the service, which becomes a “monitored service”.

In the presence of a high number of services, the computing and storage of metrics can be a high-demanding task, specially if it is done in a centralized manner. Consequently, the monitoring task must be as decentralized and low-intrusive as possible. Our design considers one *Monitoring* component attached to each monitored service, that collects information from it, and exposes an interface to obtain the computed metrics. This approach is decentralized and specialized with respect to the monitored service. On the other side, some metrics may require additional information from other services: for example, to compute the cost of running a composition, the *Monitoring* component would require to know the cost of all the services used while serving some request. To address this situation, the *Monitoring* component is capable of connecting to the *Monitoring* components of other services. This way, the set of *Monitoring* components are inter-connected forming an architecture that reflects the composition of the monitored services and forming a “monitoring backbone” as shown in Figure 3. The metrics computed at each service can flow and can be used by another component.

B. SLA Analysis

The *Analyzer* component checks the compliance to a previously defined SLA. An SLA is defined as a set of simpler terms called *Service Level Objectives* (SLOs), which are represented by conditions that must be verified at runtime. The conditions may be very simple ones, f.e., triples $\langle metric, comparator, value\rangle$ expressing, for instance, “$respTime \leq 30sec$”; or more complex expressions involving other metrics or operations on them like

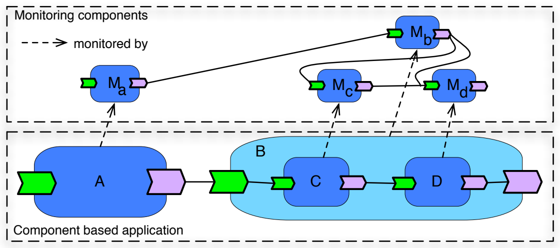


Figure 3. Monitoring layer for an SCA application

"$cost(weatherService) < 2 \times cost(mappingService)$", where the metrics used by different services are required. The *Analyzer* obtains the values of the metrics it needs from the *Monitoring* component, as exemplified in Figure 4, and thanks to the interconnected *Monitoring* components, it can obtain metrics from other services as well.

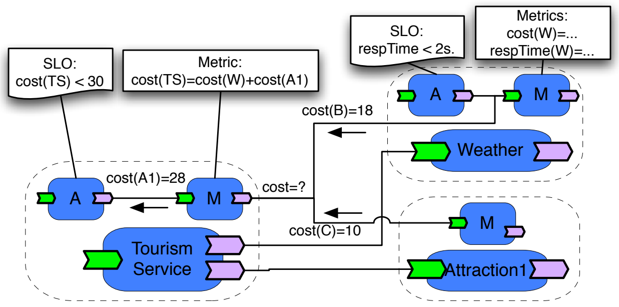


Figure 4. SCA Components with Analysis (A) and Monitor (M) components. *Tourism Service* and *Weather* have different SLAs. Metric *cost* is computed in *Tourism Service* by calling the monitors of *Weather* and *Attraction1*.

As input, the *Analyzer* receives a set of conditions (SLOs) to monitor, expressed in a predefined language. The *Analyzer* checks the compliance of all the stored SLOs according to the metrics obtained from the *Monitoring* component. The *Analyzer* checks if the SLA is being fulfilled, and if not, it sends an alarm notification through a client interface. The consequences of this alarm are out of the scope of the *Analyzer*. The *Analyzer* may also be configured in a proactive way to detect not only SLA violations, but also foreseeable SLA violations, which may be more useful in some contexts, as it can allow to take preventive actions [1].

By having the *SLA Analyzer* attached to each service, the conditions can be checked closely to the monitored service and benefit of the hierarchical composition. This way, the services do not need to take care of SLAs in which they are not involved.

C. Planning

The *Planning* component contains the strategy defined for reacting to an alarm notified by the *Analyzer* component. The implemented logic can be a very simple strategy like changing the parameter of a service, or replacing one service for another service selected from a list; or a more complex strategy that requires collecting metrics of other components in order to select a subset of services that maximizes an objective function.

As input, this component receives a notification from the *Analyzer* component indicating that some condition is not (or may not be) fulfilled. The *Planning* component executes an strategy and generates a sequence of actions that aim to take the application to an objective state. If required, it can use the *Monitoring* Component to request certain metrics. The generated actions, once again, can take a very simple form (a shellscript) or a more complex one, as a sequence of actions described in a domain-specific language that can be interpreted and executed by a set of actuators.

This encapsulation allows the framework to replace at runtime the strategy to reach the objective, for example from a cost-optimizing planner to an energy-efficiency planner, or well taking no action at all.

D. Execution

The *Execution* component carries on the decided modifications to the service, or to a set of services, as indicated by the *Planning* component.

The execution requires an integrated means to access the managed service in order to execute the actions upon it. In a similar way to the *Monitoring* component, the *Execution* component must implement any specific protocol required by the managed service in order to be able to trigger adaptations on it.

The set of actions demanded may involve not only the managed service, but also different service(s). For this reason, the *Execution* component is also able to communicate with the *Execution* components attached to some other components and send actions to them as part of the main reconfiguration action. The set of connected *Execution* components forms an "execution backbone" that propagates the actions from the component where the actions have been generated to each of the specific components where some part of the actions must take place, possibly hierarchically down to their respective inner components. This approach allows to decentralize the execution of the actions.

One of the challenges in the execution phase is to ensure that the reconfiguration or adaptations actions will not make the application enter in an unsafe state. This problem is left to the execution implementation.

Figure 5 shows a sample situation in which the *Analyzer* component of the *Tourism Service* component finds out that the cost of the compostion if exceeding a desired threshold. The *Analyzer* notifies the *Planning* component, which executes an strategy oriented to replace the component with the higher cost by a cheaper one. The *Planning* component uses its *Monitoring* component to get the cost metric of several components, and decides to replace the component *Attraction1* by the (functionally equivalent) *Attraction2*. The action is carried on by the *Execution* component.

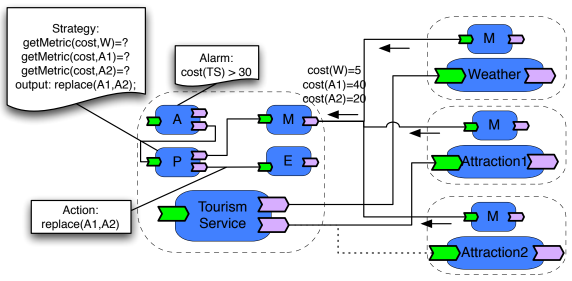


Figure 5. SCA Components Tourism Service reacts to an SLA violation by replacing the component that features the highest cost.

IV. Implementation

This section describes our prototype implementation over the GCM model. We describe the pieces of the framework that we have implemented and exemplify how they can be used to provide self-adaptability in the context of the scenario described in Section II.

A. Background: SCA compliant GCM/ProActive

The ProActive Grid Middleware [2] is a Java middleware, which aims to achieve seamless programming for concurrent, parallel and distributed computing, by offering an uniform active object programming model, where these objects are remotely accessible via asynchronous method invocations with futures. Active Objects are instrumented with MBeans, which provide notifications about events at the implementation level, like the reception of a request, and the start and end of a service. The notification of such events to interested third parties is provided by an asynchronous and grid enabled JMX connector.

The Grid Component Model (GCM) [3] is a component model for applications to be run on computing grids, that extends the Fractal component model [4]. Fractal defines a component model where components can be hierarchically organized, reconfigured, and controlled offering functional server interfaces and requiring client interfaces (as shown in Figure 6). GCM extends that model providing to the components the possibility to be remotely located, distributed, parallel, and deployed in a grid environment, and adding collective communications (multicast and gathercast interfaces). In GCM it is possible to have a componentized membrane [5] that allows the existence of non-functional (NF) components, also called *component controllers* that take care of non-functional concerns. NF components can be accessed through NF server interfaces, and components can make requests to NF services using NF client interfaces (shown respectively on top and bottom of C in Figure 6).

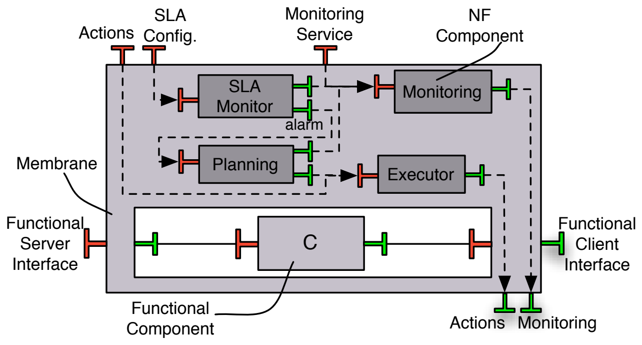


Figure 6. Framework implementation weaved to a primitive GCM component C

The use of NF components instead of simple object controllers as in Julia, the Fractal reference implementation, allows to have a more flexible control of NF concerns and to develop more complex implementations, as the NF components can be bound to other NF components within a regular component application. In this sense, this paper complements some previous ones about the componentized membrane [3], [5], [6], particularly addressing the concerns of self-adaptability in service-oriented contexts.

GCM/ProActive is the reference implementation of GCM, within the ProActive middleware, where components are implemented by Active Objects, which can be used to implement new services, or wrap existent legacy applications like C/Fortran MPI code, or a BPEL code.

The GCM/ProActive platform provides asynchronous communications with futures between bound components through GCM bindings. GCM bindings are used to provide asynchronous communication between GCM components, and can also be used to connect to other technologies and communications protocols, like Web Services, by implementing the compliance to these protocols via specific controllers in the membrane. These controllers have been used to allow GCM to act as an SCA compliant platform, in a similar way as achieved by the SCA FraSCAti [7] platform, which however bases upon non distributed components (Fractal/Julia) in contrary to GCM ones.

B. Framework Implementation

The framework is implemented in the GCM/ProActive middleware as a set of NF components that can be added or removed at runtime to the membrane of any GCM component, which becomes a managed service of the application. The ability to reconfigure the composition of the membrane at runtime is provided by the middleware [5].

We have designed a set of predefined components that implement each one of the elements we have described in Section III. This is just one of possible implementations, and particularly this has been designed to provide self-adaptable capabilities to the composition.

The general implementation view for a single GCM component is shown in Figure 6 (using the GCM graphical notation [3]), and resembles the design presented in Figure 2. The framework is weaved in the primitive GCM component *C* by inserting NF components in its membrane. Monitoring and management features are exposed through the NF server interfaces “Monitoring Service”, “SLA Config” and “Actions” (top of Figure 6). NF components can communicate with the NF components of other GCM components through the NF client interfaces “Monitoring” and “Actions” (bottom of Figure 6). The sequence diagram of the self-adaptability loop is shown in Figure 7.

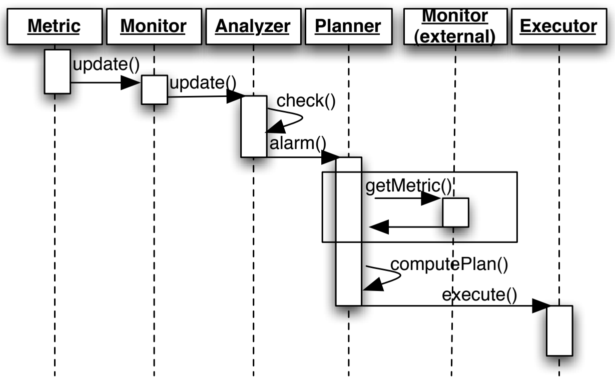


Figure 7. Sequence diagram for the autonomic control loop

C. Monitoring

We have designed a set of probes, f.e. for CPU load and memory use, and incorporated them along with the events produced by the GCM/ProActive platform [8]. Over them, we provide a *Monitoring* component, shown on Figure 8, which includes (1) an *Event Listener* that receives events from a GCM component and provides a common ground to access them; (2) a *Record Store* to store records of monitored data that can be used for later analysis; (3) a *Metric Store* that stores objects that we call *Metrics*, which actually compute the desired metrics using the records stored, or the events caught; and (4) a *Monitor Manager*, which provides the interface to access the stored metrics, and add/remove them to/from the Metrics Store.

The *Monitor Manager* receives a Metric as a Java object with a *compute* method, and inserts it in the *Metric Store*. The *Metric Store* provides to the Metrics the connection to the sources that they may need; namely, the *Record Store* to get already sensed information, the *Event Listener* to receive sensed information directly, or the *Monitoring* component of other external components, allowing access to the distributed set of monitors (i.e., to the monitoring backbone). For example, a simple *respTime* metric to compute the response time of requests, requires subscription to the *Event Listener* for events related to the start and finish times of the service of a request.

As a more complex example, the Tourism Service needs to know the decomposition of the time spent while serving a specific request r_0. For this, a metric called *requestPath* for a given request r_0 can ask the *requestPath* to the *Monitoring* components of all the services involved while serving r_0, which can repeat the process themselves; when no more calls are found, the composed path is returned with the value of the *respTime* metric for each one of the services involved in the path. Once the information is gathered in the *Monitoring* component of the Tourism Service, the complete path is built and it is possible to identify the time spent in each service.

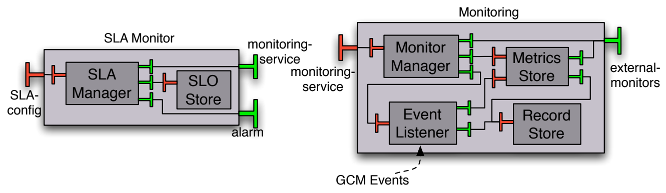


Figure 8. Internal Composition of the Monitoring component (right) and the SLA Monitor component (left)

D. SLA Monitor

The *SLA Monitor* is implemented as a component that queries the *Monitoring* component. The *SLA Monitor* consists in (1) an *SLA Manager*, which exposes an interface that allows to add/remove SLOs expressed in a specific format, checks the fulfillment of the SLOs, and sends a notification when some of them are not fulfilled; and (2) an *SLO Store*, which maintains the list of SLOs. The composition is shown in the left side of Figure 8.

In this implementation, an SLO is described as a triple $\langle metricN, comparator, value\rangle$, where *metricN* is the name of a metric. The SLA Monitor subscribes to the *metricN* from the *Monitoring* component to get the updated values and check the compliance of the SLO.

For example, the Tourism Service service includes the SLO: “All requests must be served in less than 30 secs”, described as $\langle respTime, <, 30\rangle$. The *SLA Manager* receives this description and sends a request to the *Monitoring* component for subscription to the *respTime* metric. The condition is then stored in the *SLO Store*. Each time an update on the metric is received, the *SLA Manager* checks all the SLOs associated to that metric. In case one of them is not fulfilled, a notification is sent, through the *alarm* interface including the description of the faulting SLO.

E. Planning

The *Planning* component, shown on the left side of Figure 9, includes a *Strategy Manager* that receives an alarm message and, depending on the content of the alarm, it triggers one of several bound *Planner* components. Each one of the *Planner* components implements the logic required to make a decision to restore the state of the application to comply with a failed SLO. Each *Planner* component can access the *Monitoring* components to retrieve any additional information they may need about the composition; the output is expressed as a list of actions in a predefined language.

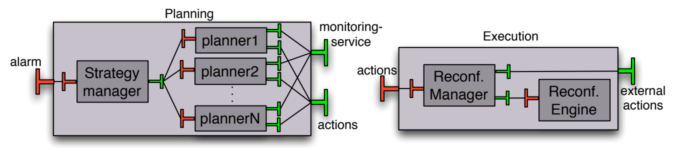


Figure 9. Internal Composition of the Planning and the Execution components

In our implementation we profit of selective 1-to-N communications provided by GCM to decide which *Planner* component to trigger. For example, if the SLO violated is related to response time, we may trigger a performance-oriented recomposition; or if a given cost has been surpassed, we may trigger a cost-saving algorithm. The decision of what strategy to use is taken in the *Strategy Manager* component. However, the possibility of having multiple strategies might be a source for conflicting decisions; while we do not provide a method to solve these kind of conflicts, we assume that the conflict resolution behaviour, if required, is provided by the *Strategy Manager*.

We have implemented a simple planning strategy that, given a particular request, asks to compute the *requestPath* for that request, then finds the component most likely responsible for having broken the SLO, and then creates a plan that, when executed, will replace that component for another component from a set of possible candidates. Applied to the Tourism Service, suppose a request has violated the SLO $\langle respTime, <, 30\rangle$. The *Strategy Manager* activates the *Planner* component that obtains the *requestPath* for that request along with the corresponding response time, selects the component that has taken the highest time, then obtains a set of possible replacements for that component and obtains for each of them the *avgRespTime* metric. The output is a plan expressed in a predefined language that aims to replace the slowest component by the chosen one.

Clearly this strategy does not intend to be general, and does not guarantee an optimal response in several cases. Even, in some situations, it may fail to find a replacement and in that case the output is an empty set of actions. However, this example describes a planning strategy that can be added to implement an adaptation for self-optimizing and that uses monitoring information to create a list of actions.

F. Execution

The *Execution* component, shown on the right side of Figure 9, includes a *Reconfiguration Engine*. This engine uses a domain specific language called PAGCMScript, an extension of the FScript [9] language (designed for Fractal components), which supports GCM specific features like distributed location, collective communications, and remote instantiation of components.

The *Execution* component receives actions from the *Planning* component. As many strategies may express actions using different formats, a component called *Reconfiguration Manager* may need to apply a transformation to express the actions in an appropriate language for the *Reconfiguration Engine*. The *Reconfiguration Manager* may also discriminate between actions that can be executed by the local component, or those that must be delegated to external *Execution* components.

In the example, once the “Attraction2” provider has been selected, it can be unbound from the “Tourism Service” using a PAGCMScript command like the following, whose effect can be seen in Figure 5:

```
unbind($tourism/interface::"attraction2")
```

G. Generalization

As GCM is an SCA compliant platform, the GCM-based framework, as shown in Figure 6 can be described in SCA terms providing a view that can be realized for any SCA runtime platform like that in Figure 2. The deployment of the framework may be done by injecting the required SCA description in the SCA ADL file. However, in order to allow this modification to occur at runtime, we have provided a console application that can use the standard non-functional API of GCM components to insert or remove at runtime the required components of the framework.

The console, while not being itself a part of the framework, shows that an external application can be built and connected to the NF interfaces of the running application and handle at runtime the composition and any subsequent reconfiguration, if needed, of the monitoring and management framework itself.

V. Related Work

Several works exist regarding monitoring and management of service-oriented applications. Most of them tackle separately monitoring infrastructures [10], SLA monitoring and analysis [11], SLA fulfillment [1], and planning strategies for adaptation [12], [13], [14]. A few others, like us, propose a complete framework. The work [15] is similar to ours in that they propose a generic context-aware framework that separates the steps of the MAPE control loop to provide self-adaptation; their work allows the implementation of self-adaptive strategies, though not much is mentioned about runtime reconfigurability, or the possibility to have multiple strategies. Also, we do not necessarily consider that all services require the same level of autonomicity.

CEYLON [16] is a service-oriented framework for integrating autonomic strategies available as services and use them to build complex autonomic applications. They provide the managers that allow to integrate and adapt the composition of the autonomic strategies according to evolving conditions. In CEYLON, autonomicity is a main *functional* objective in the development of the application, while in our case, we aim to provide autonomic QoS-related capabilities to already existing service based applications. Also, we take benefit of the business-level components intrinsec distribution and hierarchy to split the implementation of monitoring and management requirements across different levels, thus enforcing scalability.

VI. Conclusions and Perspectives

We have presented a generic component-based framework for supporting monitoring and management tasks of component-based SOA applications. The component based approach allows a clear separation of concerns between the functional content and the management tasks. We have implemented a prototype that provides a self-adaptation loop for component-based services, thanks to the composition of appropriate monitoring, SLA management, planning and reconfiguration components. This prototype has been developed in the context of an SCA compliant platform that includes dynamic reconfiguration and distribution capabilities.

This approach provides a high degree of flexibility as the skeleton we have provided for the autonomic control loop can be personnalized to e.g., support different planning strategies, and leverage heterogeneous monitoring sources to provide the input data that these strategies may need (for example, performance, price, energy consumption, availability). Early evaluation shows a small overhead no bigger than 4% in the execution of basic reconfiguration operations (namely, insertion of a new SLO and metrics, communication between functional and non-functional components, and runtime architectural rebindings), with respect to the performance before attaching the components of the framework. We expect to provide a set of benchmarks to clearly establish this overhead. The hierarchical approach is expected to provide high scalability, though a bigger experimentation set is still required.

References

[1] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar, and F. Leymann, "Runtime prediction of service level agreement violations for composite services," in *Proceedings of the 2009 international conference on Service-oriented computing*, ser. ICSOC/ServiceWave'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 176–186.

[2] "ProActive Parallel Suite," accessed on 24-Mar-2011. [Online]. Available: http://proactive.inria.fr/

[3] F. Baude, D. Caromel, C. Dalmasso, M. Danelutto, V. Getov, L. Henrio, and C. Pérez, "GCM: a grid extension to Fractal for autonomous distributed components," *Annals of Telecommunications*, vol. 64, no. 1-2, pp. 5–24, 2009.

[4] E. Bruneton, T. Coupaye, M. Leclercq, V. Quma, and J.-B. Stefani, "The fractal component model and its support in java," *Software: Practice and Experience*, vol. 36, no. 11-12, pp. 1257–1284, 2006.

[5] F. Baude, L. Henrio, and P. Naoumenko, "Structural reconfiguration: An autonomic strategy for gcm components," in *Proceedings of the 2009 Fifth International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 123–128.

[6] M. Aldinucci, S. Campa, P. Ciullo, M. Coppola, M. Danelutto, P. Pesciullesi, R. Ravazzolo, M. Torquati, M. Vanneschi, and C. Zoccolo, "A framework for experimenting with structured parallel programming environment design," in *Parallel Computing - Software Technology, Algorithms, Architectures and Applications*, ser. Advances in Parallel Computing. North-Holland, 2004, vol. 13, pp. 617 – 624.

[7] L. Seinturier, P. Merle, D. Fournier, N. Dolet, V. Schiavoni, and J.-B. Stefani, "Reconfigurable sca applications with the frascati platform," *Services Computing, IEEE International Conference on*, vol. 0, pp. 268–275, 2009.

[8] C. Ruz, F. Baude, and B. Sauvan, "Enabling SLA Monitoring for Component-Based SOA Applications – A Component-Based Approach," in *Proceedings of the Work in Progress Session SEAA 2010*. Johannes Kepler University Linz, 2010, pp. 41–42.

[9] P.-C. David, T. Ledoux, M. Lger, and T. Coupaye, "Fpath and fscript: Language support for navigation and reliable reconfiguration of fractal architectures," *Annals of Telecommunications*, vol. 64, pp. 45–63, 2009.

[10] A. Van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst, "Continuous Monitoring of Software Services: Design and Application of the Kieker Framework," p. 26, 2009.

[11] A. Michlmayr, F. Rosenberg, P. Leitner, and S. Dustdar, "Comprehensive QoS monitoring of Web services and event-based SLA violation detection," in *Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing*, ser. MWSOC '09. New York, NY, USA: ACM, 2009, pp. 1–6.

[12] G. Canfora, M. Di Penta, R. Esposito, F. Perfetto, and M. Villani, "Service composition (re)binding driven by applicationspecific qos," in *Service-Oriented Computing ICSOC 2006*, ser. Lecture Notes in Computer Science, A. Dan and W. Lamersdorf, Eds. Springer Berlin / Heidelberg, 2006, vol. 4294, pp. 141–152.

[13] V. Cardellini and S. Iannucci, "Designing a broker for qos-driven runtime adaptation of soa applications," *Web Services, IEEE International Conference on*, vol. 0, pp. 504–511, 2010.

[14] C. Ghezzi, A. Motta, V. Panzica, L. Manna, and G. Tamburrelli, "QoS Driven Dynamic Binding in-the-many," *QoSA 2010*, pp. 68–83, 2010.

[15] F. Andre, E. Daubert, and G. Gauvrit, "Towards a generic context-aware framework for self-adaptation of service-oriented architectures," *Intl. Conf. on Internet and Web Applications and Services, ICIW*, vol. 0, pp. 309–314, 2010.

[16] Y. Maurel, A. Diaconescu, and P. Lalanda, "CEYLON: A Service-Oriented Framework for Building Autonomic Managers," *7th IEEE Intl. Conf. and Workshops on Engineering of Autonomic and Autonomous Systems*, pp. 3–11, Mar. 2010.

Introduction of Self Optimization Features in a Selfbenchmarking Architecture

El Hachemi Bendahmane*‡ and Bruno Dillenseger*
*France Telecom RD, Orange Labs
Grenoble, France
Email: {Elhachemi.bendahmane,bruno.dillenseger}@orange-ftgroup.com

Patrice Moreaux‡
‡LISTIC, University of Savoie
Annecy, France
Email: patrice.moreaux@univ-savoie.fr

***Abstract*—Benchmarking client-server systems involve complex, distributed technical infrastructures, whose management deserves an autonomic approach. It also relies on observation, analysis and feedback steps that closely matches the autonomic control loop principle. While previous work have shown how to introduce autonomic load testing features through self-regulated load injection, this paper sketches the path to full self-benchmarking, introducing self-optimization features to get meaningful results. Our contribution is twofold: completion of a component-based architecture, combining several autonomic control loops, to fully support self-benchmarking, and an original constraints programming-based optimization algorithm. The relevance of this work in progress is partially evaluated through first experimental results.**

***Keywords*-benchmarking; autonomic computing; self-optimization; constraint satisfaction problem.**

I. Introduction

A. Introduction to Benchmarking

From a general point of view, the goal of benchmarking practices is to compare alternative elements or processes through a performance rating based on well defined metrics. In the field of Information Technologies and Networks, benchmarks aims at guiding the selection of different implementations and configurations of software and hardware from a performance viewpoint. For instance, one may be interested in comparing the performance of a number of Java Virtual Machines (JVM), databases, application servers, etc. Benchmarks are basically specifications, edited by organizations like the Standard Performance Evaluation Corporation [1] or the Transaction Processing Performance Council [2]. Some benchmarks, like RUBiS [3][4] in the field of web application servers, also publish a reference implementation.

Benchmarking a client-server system typically takes:

- a reference application that uses the scoped alternative elements;
- a workload specification defining the flow of requests submitted to the reference application;
- a set of performance metrics of interest (e.g., application response times or computing resources usage).

For example, RUBiS provides an on-line auction web application, implemented with different design patterns, that can be run on a variety of JVMs, applications servers and databases. RUBiS also provides an HTTP traffic injection utility that defines a special mix of different HTTP requests. By measuring the performance of this application with different elements, the tester can compare performance of these alternatives and make the best technological choices.

B. Benchmarking and Optimization

One of the key issues with benchmarking is to get meaningful, actually comparable measures. As a consequence, all alternatives must be tuned, or, in other words, optimally parametrized to reach their best performance. Then, the benchmark is used to rate and compare possible parametrizations and find the optimal set of parameters values for the whole set of involved elements.

To go on with the RUBiS example, JVMs, applications servers and databases all have specific parameters whose settings may (or may not) influence the overall application performance. Let us mention heap size, garbage collection policy, size of thread pools, size of database connection pools, size of caches, etc. For example, [5] provides a tuning guide for Java EE applications, giving the main parameters of interest for performance. Of course, optimal settings of one alternative are likely to depend on the application. They also depend on the other elements' settings in the whole system.

To conclude, benchmarking activities typically relies on looping on the following steps:

- setting the system elements' parameters values to improve the system performance (tuning),
- benchmarking the system to rate the performance of this new parametrization.

Our idea is to develop an autonomic computing approach to benchmarking [6]. The full vision encompasses both the benchmarking step and the tuning step. This combination comes with two major interests: automation of benchmarking campaigns, and pre-optimization of a system before its actual production use.

C. Self-Regulated Load Injection

For the autonomic benchmarking step, we reuse previous work featuring Self-Regulated Load Injection (SRLI), as published in [7]. SRLI consists in an autonomous exploration of the system performance under a growing load injection. SRLI is based on a control loop enabling a smart workload increase according to the evolution of relevant metrics. SRLI stops as soon as some given saturation criteria, such as service response time or server's CPU usage, passes given thresholds.

As common load injection tools, SRLI defines the workload level in terms of number of active virtual users. One virtual user represents an elementary sequence of requests and think times, typically representing a real user session. Starting from a single virtual user, SRLI adds virtual users until reaching a saturation threshold. Finally, SRLI delivers a number of metrics such as average response time or server throughput. But the most relevant metric for our optimization purpose is the maximum number of virtual users that could be run before reaching saturation. As a matter of fact, it meters the maximum server's capacity under given operating and quality of service constraints.

On the software architecture side, SRLI is built as a component-based system, according to the Fractal model [8]. It springs from research work aiming at providing a component-based framework for building autonomic systems [9], as well as from the CLIF component-based load testing framework [10].

D. Introducing self-optimization

This paper is mainly dedicated to self-optimization aspects. It also sketches its integration with SRLI to provide the full self-benchmarking vision, keeping the component-based architectural design. Our approach to self-optimization is based on a classical generate and evaluate process: choose a set of parameter values and apply them to the system under test, and then, evaluate the performance resulting from these new settings, with SRLI. This actually introduces a new control loop that we call the optimization loop.

This approach must cope with a number of issues:

- gracefully handle multiple autonomic control loops;
- possible constraints between parameters' values;
- the huge combinatorial effect between the possible values of all the parameters;
- duration of each SRLI run, which is typically several minutes for our sample web application;
- automation of parameter values change.

E. A constraint solving approach

Besides this optimization control loop principle, the peculiarity of this work is to represent the optimization problem as a Constraint Satisfaction Problem (CSP) [11]. CSP is a convenient support for representing parameters' possible values and possible constraints between parameters. It is also a convenient way of generating valid parametrizations and submit them to SRLI's evaluation.

F. Paper outline

This paper is organized as follows: section II shows how we introduce the optimization control loop and combine it with SRLI. In Section III, we detail the algorithmic aspects of our self-optimization approach, including tools, strategies and heuristics. Section IV presents our first experimental results with a Java EE application use case. Finally, we conclude in Section V, and give some open questions and perspectives.

II. Introducing Self-Optimization

A. A component-based approach

Autonomic computing [12] springs from the observation that today's Information Technologies and Networks (IT&N) systems have reached such a complexity level, that their management reaches the limits of human capabilities. In a way, autonomic computing consists in using part of the IT&N power to handle its highly complex management. Now, a trap would be to fight complexity by adding even more complexity.

As mentioned in section I, SRLI has been designed according to a general component-based approach to building autonomic systems [9]. This approach advocates for a strong architectural requirement, to cope with this possible paradox of autonomic computing. The challenge is to limit and overcome the complexity of autonomic features, and keep a full control on them. The approach consists in having a uniform component-based system representation. Autonomic control loops are themselves built as components, which provides a comprehensive and self-aware architecture.

B. Architecture of Self-Regulated Load Injection

The initial idea behind SRLI is to consider that benchmarking activities are relevant use cases for this architectural approach to building autonomic systems, because of the high complexity level of load testing infrastructures. SRLI's goal is to automatically and quickly find the performance limits of an arbitrary system. [7] shows its practical use for testing a multi-tier Java EE application, and finding its performance limits in less than 10 minutes.

SRLI's top level components are:

- *load injectors*, in charge of generating workload on the tested system, and measuring requests response times and throughput;
- *probes*, responsible for monitoring usage of computing resources (processor, memory, network…);
- one *supervisor*, giving a central access point to control and monitor all probes and load injectors;
- one *load controller*, adjusting the injected workload, in terms of number of virtual users, according to performance measures;
- one *saturation controller* checking whether saturation criteria are met or not.

SRLI combines two control loops:

1) the load injection control loop, adjusting the number of virtual users according to the response times and throughput observations. It involves the load injectors, the supervisor, and the load controller;
2) the saturation control loop, in charge of stopping the load injection control loop when some saturation thresholds are reached. It involves the probes, the supervisor, and the saturation controller.

Both loops have explicit control on each other: loop 1 first launches loop 2, and then loop 2 stops loop 1. Loop 1 is launched by external control e.g., by a user, to restart the process on a new system configuration.

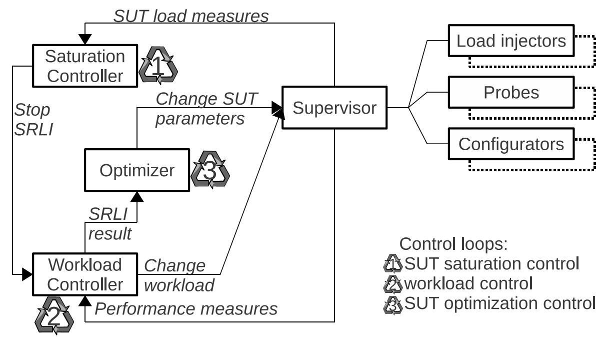


Figure 1. Global self-optimization architecture with three control loops

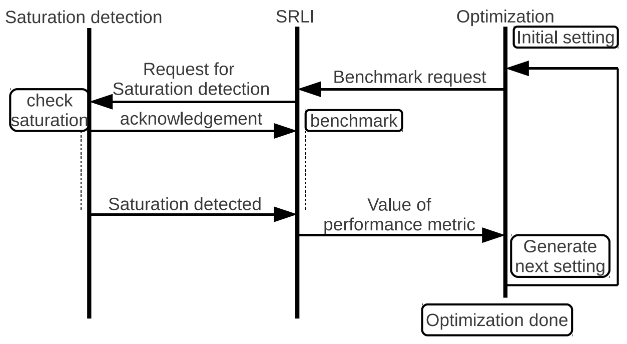


Figure 2. Sequence diagram of full autonomic benchmarking

C. Introducing our self-optimization components

Our self-optimization control loop is built from the following components:

- an *optimizer* component in charge of generating a new parametrization according to the performance metrics values it has gathered during the previous SRLI completions;
- *configurator* components in charge of applying the new parameters' values to the benchmarked system.

Section III gives details about an optimizer component based on constraint solving paradigm. Configurator components must typically stop the system under test, apply the new parametrization, and restart the system. Parameters may be set through some management protocols, such as SNMP [13] or JMX [14], or by updating configuration files. Figure 1 shows the global architecture combining SRLI and self-optimization components. The logical sequence between the three control loops' activities is depicted by figure 2.

D. Focus on control loops coordination

This global self-benchmarking approach is a use case for multiple control loops coordination. There exists three main schemes:

- explicit control between control loops, as it is currently implemented in SRLI: control loops are aware of each other;
- hierarchical control i.e., low-level control loops don't know each other but are controlled by a higher level control loop;
- implicit control i.e., control loops don't know each other but communicate through their environment.

In this work in progress, we currently use the straightforward explicit control scheme. However, we are interested in exploring the implicit control approach, in order to be able to combine some, but not necessary all, of these three control loops, possibly with other control loops. As a noteworthy example, we may be interested in keeping saturation detection and self-optimization, while replacing our massive load injection by a thin load injector checking quality of service. This way, we could reuse some of our components to support self-optimization in a production environment instead of a benchmarking environment.

III. Algorithmic aspects of Self optimization - A first attempt

In this section we explain how we find the values of the parameters required to configure the system under test to get the best behavior for a given metric m.

A. Coping with the complexity of system tuning

As mentioned in the introduction, there are a lot of parameters to set for tuning a system like an application running on a Java EE server. To manage these parameters, we can define two sets of parameters classes. The first set is related to the *software level*, that is in what part of the hardware/software stack is located the parameter. We can distinguish four classes: OS and hardware parameters, JVM parameters (memory, size of the heap, ...), Java EE or other server parameters (if this applies), Application parameters. Another set of parameters is related to the *technical level*, that is when (with regard to the running state) we can/must change the value of the parameter. We have defined three classes: before deployment of the application, before starting the application server (if this applies), when the application runs. Note that, to modify the value of a parameter, several technical means may be used such as files and JMX interactions.

Another classification of the parameters relies on the metric m. To devise efficient algorithms, it is necessary to establish how m varies with each parameter p_i, for fixed values of the other parameters. Let us denote by $\bar{p} = (p_1, \ldots, p_n) \in D = \prod_{i=1}^{n} D_i$ the vector of parameters we control for tuning the system. Based on our previous works on self-benchmarking, we assume that the function $f(\bar{p}) = m(\bar{p})$ is concave on D. So, if we denote by f_i the "projection" functions of f on D_i, that is: $f_i(p_i) = f(\bar{p})$ for a given $(n-1)$tuple of parameters $p_j (j \neq i)$, we know that f_i has a unique maximum (at $p_m \in D_i$). This specific property of the metric m allows us to devise an adapted algorithm. The second classification of the parameters gives the relative impact of each parameter with regard to m. These information are provided by human experts and from models of the system studied if they are available.

The above classes of parameters allows us to restrict the set of parameters to n parameters $p_1, p_2, \ldots, p_n$, each one with a domain $D_i = [min_i, \ldots, max_i]$. Note that we can restrict each D_i to a finite set of integers. In our example (see Section IV), we have found at least 400 parameters that we can control. From these, we selected 15 significant parameters and in this paper, we present first results with two parameters.

B. Finding the best parameters

The problem is then to find $p^{opt} = argmax_{\bar{p} \in D}\{m(\bar{p})\}$. The main difficulty to find p^{opt} is that the computation of $m(\bar{p})$, which corresponds to a self-benchmarking experiment takes several minutes. Hence, we have to devise an algorithm requiring as few as possible such computations.

To reduce the number of calls to m, we first use the context of Constraint Satisfaction Programming (CSP) (see [11] for a presentation of CSP). The main interests of CSP for our problem is the ability to define parameters, their domains and their constraints in a declarative way and to check at runtime if a given parameter tuple is an admissible solution or not. Among the available solver tools, we have chosen the Choco [15] open source library. Choco provides several Java libraries to define the CSP and to verify the constraints. Obviously, Choco embeds also a customizable solver, but in this paper we did not use it, but in contrast we use our specific traversal algorithm in the domain D.

Algorithm 1 is based on a kind of binary search method. For each parameter p_i, we define a granularity g_i which corresponds to the minimal distance we require between two values of p_i. Then, we define the norm of a vector $\bar{p}$ as $||\bar{p}|| = \sum_{i=1}^{n} |\frac{p_i}{g_i}|$, so that we can compute the relative variation of $\bar{p}$ between $\bar{p}_c$ and $\bar{p}_p$ as $var(\bar{p}) = \frac{||\bar{p}_c - \bar{p}_p||}{||\bar{p}_p||}$. We also set the precision x of the search and the maximal number k_{max} of the main iteration. Each iteration of the algorithm computes the vector $\bar{p}_c^{opt}$, starting from its first component, and updating successively its n components. For a given component i, the function findbest computes the value of p_i which maximizes $m(\bar{p}|i,x) = m((p_1,\ldots,p_{i-1},x,p_{i+1},\ldots,p_n))$ findbest selects the values to be used to compute the metric m in a "dichotomous way", with the idea that, based on previous computations of m, we can skip some new computations. This approach gave interesting results during our first experiments (see Section IV).

Algorithm 1 Compute the optimal parameters

```
Define the initial value of $\bar{p}_c^{opt}$
$\bar{p}_p^{opt} = (1+2y)\bar{p}_c^{opt}$
$k = 1$
**while** $(var(p^{opt}) > y)$ and $(k \leq k_{max})$ **do**
  **for** $i = 1$ **to** $n$ **do**
    $(p, m) = findbest(i, \bar{p}_c^{opt}, min_i, max_i)$
    $p_{c,i}^{opt} = p$
  **end for**
  $k = k + 1$
**end while**
**return** $\bar{p}_c^{opt}$

**Function (int, double)** $findbest(i, \bar{p}, min, max)$
**while** $(max - min \geq g_i)$ **do**
  $mid = (min + max)/2$
  compute $m_{i,min} = m(\bar{p}|i, min)$
  compute $m_{i,mid} = m(\bar{p}|i, mid)$
  **if** $m_{i,min} > m_{i,mid}$ **then**
    **return** $findbest(i, \bar{p}, min, mid)$
  **else**
    compute $m_{i,max} = m(\bar{p}|i, max)$
    **if** $m_{i,mid} < m_{i,max}$ **then**
      **return** $findbest(i, \bar{p}, mid, max)$
    **else**
      $v_1 = findbest(i, \bar{p}, min, mid)$
      compute $m_{i,v_1} = m(\bar{p}|i, v_1)$
      **if** $m_{i,v_1} < m_{i,mid}$ **then**
        $v_2 = findbest(i, \bar{p}, mid, max)$
        compute $m_{i,v_2} = m(\bar{p}|i, v_2)$
        **if** $m_{i,v_1} > m_{i,v_2}$ **then**
          **return** $(v_1, m_{i,v_1})$
        **else**
          **return** $(v_2, m_{i,v_2})$
        **end if**
      **else**
        **return** $(v_1, m_{i,v_1})$
      **end if**
    **end if**
  **end if**
**end while**
**return** $(mid, m_{i,mid})$
```

IV. First results with a Java EE application server

A. Use case: JOnAS 5

JOnAS [16] is an open source, Java EE 5 certified, application server provided by the OW2 consortium. JOnAS is based on an OSGi framework, which provides mechanisms to dynamically change bundles' configuration (start, stop, reconfigure, etc.). All JOnAS components are packaged as bundles and the full JOnAS profile comes with more than 250 bundles. Therefore most Java EE-certified JOnAS services (persistence, EJB, resources . . .) are available as OSGi services to all OSGi bundles deployed on JOnAS.

JOnAS administration is performed through Java JMX [14], either with graphical tools such as JOnAS' web console (JonasAdmin) or common Java's jconsole utility, or with a command line tool like MBeanCmd provided by the JASMINe open source project [17].

For our experiment, we need to stress a benchmark application. We have chosen to deploy the simple e-commerce web application MyStore 2.0.2 on JOnAS. This sample application is available from the JASMINe project repository. This is a very simple application that does not use a database.

B. Tuning points

Parameters may be set in configuration files or through JMX if they are available as MBean attributes. In the first case, changing a parameter value requires to modify a configuration file, and most probably restart JOnAS or at least one of its services. In the latter case, a simple JMX call to JOnAS' embedded JMX server allows for changing a parameter value.

JOnAS 5 comes with 25 services and 48 configuration files, resulting in one hundred parameters. A parameter definition may set a simple type value such as an integer or a Boolean, or a complex value such as a policy definition (cache management, garbage collector ...). In the case of a policy change, the system behavior may completely differ. So, the algorithm described in section III should be fully applied, consecutively for each policy setting. As a matter of fact, the concavity assumption can't hold with such parameter types. Note that Boolean values may be regarded either as integer values or policy definition, for they have only 2 possible values that must be both evaluated anyway.

A tuning point is a parameter which impacts "heavily" the server performance. Of course, there is no formal definition of "heavily". However, some studies (see for instance [5]) in the performance testing and optimization field show that we can grab quickly 80% of performance improvement tuning the JVM heap, the thread pools, the connection pools and the caches. The remaining 20% can be obtained tuning the EJB pools, the JMS and pre-compiling JSPs. So, based on experts' works, we can define a set of main tuning parameters.

For our experiments with MyStore, we have considered the two following tuning points:

- the size of the thread pool of the HTTP connector. It is controlled by its maximal value, noted `maxThreads`.
- the size of the Application Cache. It is also controlled by its maximal value, noted `cacheMaxSize`.

To determine the value range of parameter `maxThreads`, we did an initial experiment with the default value 200. We monitored the MBean attribute `currentThreadsCount` to observe the actual number of server's active threads. From the observed values, we decided to set `maxThreads`'s range to 20 – 200. We chose a granularity of 20 threads, as a trade-off between optimization accuracy and expected experimental time. If we applied an exhaustive exploration of this range with values obtained with the same dichotomous principle as described in section III, we would get 9 possible values.

We applied the same protocol to determine `CacheMaxSize`'s range. By default it is initialized to 10240KB. In fact, since MyStore does not use so much the cache (only a few images of reduced size are cached), we expect this value could be substantially lower. We monitored the MBean attribute `cacheSize` and found out that range 10 – 100 would be relevant. We chose a 10KB granularity. There again, an exhaustive exploration of this range with values obtained with our dichotomous principle would give 9 possible values.

Table I
EXPERIMENTS RESULTS

k	i	(p1,p2)	m	duration	max-min	results
1	1	(10,20)	177	7min25s	ok	
1	1	(55,20)	144	5mn15s	ok	
1	1	(32,20)	169	5mn30s	ok	
1	1	(21,20)	133	5mn20s	no	$p_1^{opt1} = 10$
1	2	(10,110)	131	5mn45s	ok	
1	2	(10,65)	120	5mn0s	ok	
1	2	(10,42)	177	7mn25s	ok	$p_2^{opt1} = 42$
2	1	(55,42)	159	5mn55s	ok	
2	1	(32,42)	184	8mn31s	ok	
2	1	(21,42)	144	5mn25s	no	$p_1^{opt2} = 32$
2	2	(32,110)	151	6mn25s	ok	
2	2	(32,65)	144	5mn46s	ok	
2	2	(32,42)	184	8mn31s	no	$p_2^{opt2} = 42$

Note that, even if it took some time to evaluate the boundaries of the parameters, this step allowed us to reduce the size of the couple space to be searched for optimal values.

C. Experiments and results

We use SRLI to generate an increasing load on MyStore. Each virtual user runs a simple scenario resulting from a real user session capture, enriched with random think times. SRLI's stopping criteria are defined as follows:

- the SUT's CPU usage must be less than 70%,
- the SUT's JVM heap memory usage must be less than 95%,
- the SUT's RAM usage must be less than 80%.

From a technical viewpoint, the experiment infrastructure uses three computers on a Gbit/s Ethernet network:

- the SUT runs on a server with two 2.8 GHz Xeon processors and 2 GB of RAM. CPU, JVM, network and memory probes are deployed on this server;
- load injection is run on one server with a quad-core 2.0 GHz Xeon processor and 8 GB of RAM;
- the supervisor and controller components (see figure 1) are run on a common PC, whose properties do not matter for the test.

Table I shows the results obtained. k and i denote respectively the number of the main (while) iteration in Algorithm 1 and the index (the number of the parameter) in the for loop; (p_1, p_2) gives the values of the two parameters. m reports the number of virtual users, while duration is the time required to get m. "ok" (resp. "no") in the max-min column indicates if the granularity was (resp. was not) reached during the test for the given parameter couple. We note that our algorithm provides interesting savings due to the reduced number of tests with respect to the number of couples; we actually reduce the number of saturation tests from 81 to 12. This reduction springs from the concavity assumption.

V. Conclusion and future work

Benchmarking requires tested systems to be optimized, to get meaningful results. This paper presents a global vision of autonomic benchmarking addressing this issue. This vision combines an autonomous system for measuring the maximum capacity of a system, given some operating and user experience constraints, with a self-optimization feature. We describe the global architecture, based on three component-based autonomic control loops, which explicitly coordinate with each other.

Then, the paper focuses on work in progress on self-optimization. The basic principle is to generate valid settings of parameters for the tested system, apply these settings, and to get a performance rating from the autonomic load testing loop. To practically enable this idea, we address a number of issues:

- representing domains of parameters' values, including possible constraints between one another. We propose to use Constraint Satisfaction Programming.
- limiting the huge combinatorial effect between parameters values (algorithmic complexity) and the load testing time. Assuming that the metric, as a function of all parameters, has no local maximum value, our algorithm drastically limits complexity.

Our first experimental results on a sample Java EE web application are promising. We actually needed to benchmark only 12 out of 81 valid settings, each benchmark duration being limited to an average 7min30s. Hence the global campaign lasted less than 2 hours instead of about 10 hours and 40 minutes (taking parameters setting time and system restart duration).

Our future work will be dedicated to complete our optimization algorithm implementation, and to actually integrate benchmarking and optimization, to get a fully autonomous system. We will build the global architecture with an implicit coordination between control loops, to get loosely coupled controller components.

Acknowledgment

We thank the Choco team and the JOnAS team for their valuable support. This research is supported by the French ANRT and Agence Nationale de la Recherche with the ANR-08-SEGI-017.

References

[1] "Standard performance evaluation corporation," http://www.spec.org/, checked on 18th March 2011.

[2] "Transaction processing performance council," http://www.tpc.org, checked on 18th March 2011.

[3] E. Cecchet, A. Chanda, S. Elnikety, J. Marguerite, and W. Zwaenepoel, "Performance comparison of middleware architectures for generating dynamic web content," in *Proceedings of the ACM/IFIP/USENIX 2003 International Conference on Middleware*, ser. Middleware '03. Springer-Verlag New York, Inc., 2003, pp. 242–261.

[4] "RUBiS - Rice University Bidding System," http://rubis.ow2.org/. Checked on 18th March 2011, Nov 2009.

[5] S. Haines, *Pro Java EE 5, Performance Management and Optimization*. Apress, 2006.

[6] F. Boyer, C. Taton, J. Philippe, and B. Dillenseger, "Selfware self-optimization: algorithms, architecture and design principles," Selfware Deliverable SP2-L2, http://sardes.inrialpes.fr/~boyer/selfware/documents/SP2-L2-Auto-Optimisation.pdf, Tech. Rep., june 2008, checked on 7th April 2011.

[7] A. Harbaoui, N. Salmi, B. Dillenseger, and J. Vincent, "Introducing queuing network-based performance awareness in autonomic systems," in *Proc. 2010 Sixth International Conference on Autonomic and Autonomous Systems*, ser. ICAS '10. IEEE Computer Society, 2010, pp. 7–12.

[8] G. S. Blair, T. Coupaye, and J.-B. Stefani, "Component-based architecture: the fractal initiative," *Annales des Télécommunications*, vol. 64, no. 1-2, pp. 1–4, 2009.

[9] ANR Selfware project, "Selfware: Lessons learned to build autonomic systems," http://sardes.inrialpes.fr/~boyer/selfware/documents/SP1-L3-Architecture.pdf. Checked on 7th April 2011, 2008.

[10] B. Dillenseger, "Clif, a framework based on fractal for flexible, distributed load testing," *Annales des Télécommunications*, vol. 64, no. 1-2, pp. 101–120, 2009.

[11] F. Rossi, P. van Beek, and T. Walsh, *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. New York, NY, USA: Elsevier Science Inc., 2006.

[12] J. O.Kephart and D. M.Chess, "The vision of autonomic computing," IBM Thomas J.Watson Research Center, january 2003.

[13] IETF, "A simple network management protocol (rfc 1157)," http://www.ietf.org/rfc/rfc1157.txt. Checked on 18th March 2011, May 1990.

[14] J. C. Process, "Java management extensions (jsr 3)," http://www.jcp.org/en/jsr/detail?id=3. Checked on 18th March 2011, November 2006.

[15] H. Cambazard, N. Jussien, F. Laburthe, and G. Rochart., "The choco constraint solver." in *INFORMS Annual meeting, Pittsburgh, PA, USA*, Pittsburgh, PA, USA, November 2006.

[16] OW2.org Consortium, "JOnAS Java open application server," http://jonas.ow2.org. Checked on 7th April 2011.

[17] OW2 Consortium, "JASMINe, the smart tool for your SOA platform management," http://jasmine.ow2.org. Checked on 7th April 2011.

An Introduction to Evolving Systems: Adaptive Key Component Control with Persistent Disturbance Rejection

Mark J. Balas
Electrical and Computer Engineering Department,
University of Wyoming
1000 E. University Ave., Laramie, WY 82071
mbalas@uwyo.edu

Susan A. Frost
Intelligent Systems Division,
NASA Ames Research Center
M/S 269-1, Moffett Field, CA 94035
susan.a.frost@*nasa*.gov

***Abstract*— This paper presents an introduction to Evolving Systems, which are autonomously controlled subsystems which self-assemble into a new Evolved System with a higher purpose. Evolving Systems of aerospace structures often require additional control when assembling to maintain stability during the entire evolution process. This is the concept of Adaptive Key Component Control which operates through one specific component to maintain stability during the evolution. In addition this control must overcome persistent disturbances that occur while the evolution is in progress. We present theoretical results for the successful operation of Adaptive Key Component control in the presence of such disturbances and an illustrative example.**

Keywords- adaptive control; aerospace systems.

I. INTRODUCTION

Evolving Systems [1]-[2] are autonomously controlled subsystems which self-assemble into a new Evolved System with a higher purpose. Evolving Systems of aerospace structures often require additional control when assembling to maintain stability during the entire evolution process [3]-[5]. An adaptive key component controller has been shown to restore stability in Evolving Systems that would otherwise lose stability during evolution [6]. The adaptive key component controller uses a direct adaptation control law to restore stability to the Evolving System through a subset of the input and output ports on one key component of the Evolving System. Much of the detail of Evolving Systems appears in the chapter [8]. In this paper, we will deal with the situation where persistent disturbances can appear in some components and must be mitigated by the adaptive key component controller. Such disturbances will often be attendant in actively controlled rendezvous and docking.

The control laws used by the adaptive key component controller to restore stability in an Evolving System are guaranteed to have bounded gains and asymptotic tracking if the Evolved System is almost strictly dissipative. Hence, it is desirable to know when the dissipativity traits of the subsystem components, including the key component, are inherited in an Evolving System. We present results describing when an Evolving System will inherit the almost strict passivity traits of its subsystem components. Then we will present an adaptive key component controller that restores asymptotic stability with bounded adaptive gains and mitigates the effect of persistent disturbances during evolution.

II. MATHEMATICAL FORMULATION OF EVOLVING SYSTEMS

A mathematical formulation of a nonlinear time-invariant Evolving System is given here. Consider a system of *L* components of individually, actively controlled subsystems which can be described by the following equations for the *i*th component:

$$\begin{cases} \dot{x}_i = f_i(x_i, u_i) \\ y_i = g_i(x_i, u_i) \end{cases} \tag{1}$$

where $i = 1,2,\ldots L$. The *i*th component has a Performance Cost Function J_i and a Lyapunov Function V_i. These are the building blocks of the Evolving System. When these individual components are joined to form an Evolved System, the new entity becomes:

$$\begin{cases} \dot{x} = f(x,u) \\ y = g(x,u) \end{cases} \tag{2}$$

with $x = [x_1 \ldots x_L]^T$, $y = [y_1 \ldots y_L]^T$, Performance Cost Function *J*, and Lyapunov Function *V*.. The *i*th component in the above Evolved System is given by:

$$\dot{x}_i = f_i(x_i,u_i) + \sum_{j=1}^{L} \varepsilon_{ij} f_{ij}(x_i, x_j, u_j); \quad 0 \le \varepsilon_{ij} \le 1 \tag{3}$$

where $f_{ij}(x_i, x_j, u_j)$ represents the interconnections between the *i*th and *j*th components. Note that when $\varepsilon_{ij} = 0$, the system is in component form and when $\varepsilon_{ij} = 1$, the system is fully evolved. As the system evolves, or joins together, the ε_{ij} 's evolve from 0 to 1.

The components of the Evolving System are actively controlled by means of local control. Local control means dependence only on local state or local output information, i.e., $u_i = h_i(x_i)$ or $u_i = h_i(y_i)$. In general, the local controller on the i^{th} component would have the form:

$$\begin{cases} u_i = h_i(y_i, z_i) \\ \dot{z}_i = l_i(y_i, z_i) \end{cases} \tag{4}$$

where z_i is the dynamical part of the control law. Local control will be used to keep the components stable and meet the individual component performance requirements, J_i.

Once the system is fully evolved, the i^{th} component in the fully evolved system $\varepsilon_{ij} = 1$, becomes:

$$\dot{x}_i = f_i(x_i, u_i) + \sum_{j=1}^{L} f_{ij}(x_i, x_j, u_j) \tag{5}$$

A state space version of the i^{th} individual component of an Evolving System where the components are connected through the states can be represented as:

$$\begin{cases} \dot{x}_i = A_i(x_i) + B_i(x_i)u_i \\ +\sum_{j=1}^{L} \varepsilon_{ij} A_{ij}(x_i, x_j);\ x_i(0) \equiv x_{0_i} \\ y_i = C_i(x_i) \end{cases} \tag{6}$$

where $i = 1,2,\ldots L$, $x_i \equiv [x_1^i \ldots x_{n_i}^i]^{\mathrm{T}}$ is the component state vector, $u_i \equiv [u_1^i \ldots u_{m_i}^i]^{\mathrm{T}}$ is the control input vector, $y_i \equiv [y_1^i \ldots y_{p_i}^i]^{\mathrm{T}}$ is the sensor output vector, $(A_i(x_i), B_i(x_i), C_i(x_i))$ are vector fields of dimension $n_i \text{ x } n_i$, $n_i \text{ x } m_i$, and $p_i \text{ x } n_i$, respectively, and the connection forces between components are represented in the $n_i \text{ x } n_j$ connection matrix, $A_{ij}(x_i, x_j)$ with $\varepsilon_{ji} = \varepsilon_{ij}$. The state space representation of the Evolved System then becomes:

$$\begin{cases} \dot{x} = A(x) + B(x)u \\ y = C(x) \end{cases} \tag{7}$$

which can also be written as $(A(x), B(x), C(x))$.

III. Inheritance of Subsystem Traits in Evolving Systems

We say a subsystem trait, such as stability, is inherited when the Evolved System retains the characteristic of the trait from the subsystem. Previous papers have examined the inheritance of stability and shown that stability is not a generally inherited trait [3]-[5] and [8]. Inheritance of almost strict passivity of subsystems is desirable in Evolving Systems that use an adaptive key component controller to restore stability.

In previous papers, [5]-[6], a key component controller has been proposed to restore stability to Evolving Systems which would otherwise lose stability during evolution. The design approach used by the key component controller is for the control and sensing of the components to remain local and unaltered except in the case of one key component which has additional local control added to stabilize the system during evolution. The key component controller operates solely through a single set of input-output ports on the key component, see Figure 1.

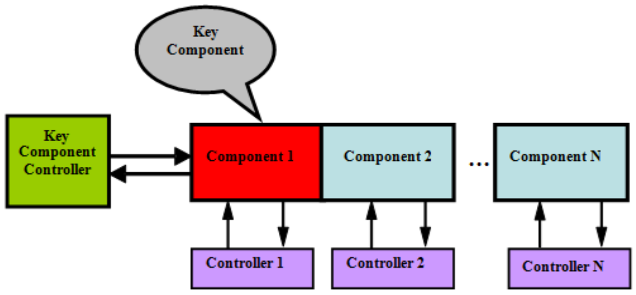


Figure 1. Key component controller.

Only the key component of the Evolving System needs modification to restore the inheritance of stability. A clear advantage of the key component design is that components can be reused in many different configurations of Evolving Systems without the need for component redesign. The reuse of components which are space-qualified, or at least previously designed and unit tested, could reduce the overall system development and testing time and should result in a higher quality system with potentially significant cost savings and risk mitigation.

In many aerospace environments and applications, the parameters of a system are poorly known and difficult to obtain. Adaptive key component controllers, which make use of a direct adaptation control law, are a good design choice for restoring stability in Evolving Systems where access to precisely known parametric values is limited. The sufficient condition for an Evolving System with an adaptive key component controller to be guaranteed to have bounded gains and to have asymptotic output tracking is that the system be almost strictly dissipative. So, we are interested in the conditions under which the inheritance of almost strict dissipativity can be guaranteed in Evolving Systems.

IV. Inheritance of Almost Strict Dissipativity in Eveloving Systems

Inheritance of almost strict dissipativity of subsystems is desirable in Evolving Systems that use an adaptive key component controller to restore stability.

Consider a ***Nonlinear System*** of the form:

$$\begin{cases} \dot{x} = A(x) + B(x)u \\ y = C(x) \end{cases}$$

We say this system is ***Strictly Dissipative*** when

$$\exists V(x) > 0 \forall x \neq 0$$

such that the Lie derivatives satisfy:

$$\begin{cases} L_A V \equiv \nabla V A(x) \leq -S(x) \ \forall x \\ L_B V \equiv \nabla V B(x) = C^T(x); \nabla V \equiv \text{gradient } V \end{cases} \tag{8}$$

The function $V(x(t))$ is called the ***Storage Function*** for (7), and the above says that the storage rate is always less than the external power. This can be seen from

$$\begin{aligned} \dot{V} &\equiv \nabla V[A(x) + B(x)u] \\ &\leq -S(x) + C^T(x)u \\ &= -S(x) + \langle y, u \rangle \end{aligned} \tag{9}$$

Taking $u \equiv 0$, it is easy to see that (9) implies (8a) but not necessarily (8b); so (8) implies (9) but not conversely. They are only equivalent if (8a) is an equality. (When equality holds in (8) and (9), the property is known as **Strict Passivity**.)

We will say a system (u, y) is **Almost Strictly Dissipative (ASD)** when there is some output feedback, $u = G_* y + u_r$, so that the following is strictly dissipative:

$$\begin{cases} \dot{x} = A_C(x) + B(x)u_r \\ A_C(x) \equiv A(x) + B(x)G_* C(x) \\ y = C(x) \end{cases} \tag{10}$$

Now if each component is ASD, then we have

$$\begin{cases} \nabla V_i[A_i(x_i) + B_i(x_i)G_i C_i(x_i)] \leq -S_i(x_i) \\ \qquad\qquad + \sum_{j=1}^{L} \varepsilon_{ij} \nabla V_i A_{ij}(x_j, u_j) \\ \nabla V_i B_i(x_i) = C_i^T(x_i); \nabla V_i \equiv gradient V_i \end{cases} \tag{11}$$

Due to the interconnection terms, (11) is not necessarily Strictly Dissipative. However, in some circumstances, the interconnection terms have a special form and ASD is inherited when the system evolves.

Suppose we have a pair of subsystems of the form:

$$\begin{cases} \dot{x}_i = A_i(x_i) + \varepsilon B_i(x_i)u_i + B_i^A(x_i)u_i^A \\ y_i = C_i(x_i) \\ y_i^A = C_i^A(x_i) \end{cases} \tag{12}$$

where $i = 1,2$ and both subsystems $\left(\begin{bmatrix} u_1 \\ u_1^A \end{bmatrix}, \begin{bmatrix} y_1 \\ y_1^A \end{bmatrix}\right)$ and $\left(\begin{bmatrix} u_2 \\ u_2^A \end{bmatrix}, \begin{bmatrix} y_2 \\ y_2^A \end{bmatrix}\right)$ have storage functions V_i. We have the following result:

Theorem 1: If the subsystems (u_1^A, y_1^A) and (u_2^A, y_2^A) are ASD and

$$\nabla V_i B_i(x_i) = C_i^T(x_i); i = 1,2 \tag{13}$$

then the resulting feedback connection, $y_1 = u_2$ and $u_1 = -y_2$, will leave the composite system $\left(u_A \equiv \begin{bmatrix} u_1^A \\ u_2^A \end{bmatrix}, y_A \equiv \begin{bmatrix} y_1^A \\ y_2^A \end{bmatrix}\right)$ almost strictly passive.

Proof: See Appendix.

In [3]-[4], it was shown that the physical connection of components is equivalent to the feedback connection of the admittance of one to the impedance of the other. Consequently, if (u_1, y_1) and (u_2, y_2) are in Admittance/Impedance form, then Theorem 1 shows that ASD is an inherited property for Nonlinear Evolving Systems.

V. Mathematical Formulation of Adaptive Key Component Controller with Persistent Disturbance Mitigation

Our <u>Key Component</u> is chosen to be <u>Component #1</u> and will be modeled by the following ***Nonlinear System with an External Persistent Disturbance***:

$$\begin{cases} \dot{x}_1 = A_1(x_1) + \varepsilon B_1(x_1)u_1 + B_1^A(x_1)u_1^A + \Gamma_1(x_1)u_D \\ y_1 = C_1(x_1) \\ y_1^A = C_1^A(x_1) \end{cases} \tag{14}$$

All vector fields in (14) will have the appropriate compatible dimensions and be smooth in their arguments with a single equilibrium point at 0 in a neighborhood U.

The persistent disturbance input vector $u_D(t)$ is N$_D$-dimensional and will be thought to come from the following Disturbance Generator:

$$\begin{cases} u_D = \Theta z_D \\ \dot{z}_D = F z_D ; z_D(0) = z_0 \end{cases}$$

where the disturbance state $z_D(t)$ is N$_D$-dimensional. Such descriptions of persistent disturbances were first used in [9] to describe signals of known form but unknown amplitude. For example, step disturbances yield $\Theta = 1$ and $F = 0$ while sinusoidal disturbances can be described by

$$\begin{cases} \Theta = \begin{bmatrix} 1 & 0 \end{bmatrix} \\ F = \begin{bmatrix} 0 & 1 \\ -\omega_D^2 & 0 \end{bmatrix} \end{cases}$$

where the frequency ω_D is known but the amplitudes are not.

We will assume that the Disturbance Generator parameter F is known. In many cases this is not a severe restriction, e.g. step disturbances. It turns out that it is better to rewrite the above in the following equivalent form:

$$\begin{cases} u_D = \Theta z_D \\ z_D = L\varphi_D \end{cases} \tag{15}$$

where ϕ_D is a vector composed of the known basis functions for the solutions of $z_D(t)$ and (L, Θ) need not be known. This can be seen from the following:

$$\begin{aligned} z_D(t) &= e^{Ft} z_D(0) \\ &= \left[\varphi_1(t), \varphi_2(t), \ldots, \varphi_{N_D}(t) \right] z_D(0) \cdot \\ &= \sum_{i=1}^{N_D} z_D^i \varphi_i(t) = L\varphi_D \end{aligned}$$

Note that L is directly related to F via its columns but not to θ . Some rearrangement of the entries in the columns of F is needed to create ϕ_D . A simple example of the above is given by the following:

$$\begin{cases} \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 1 \\ 1 \end{bmatrix} u_D \\ u_D = a \sin(\omega_D t + b) \\ \quad = a_1 \sin(\omega_D t) + a_2 \cos(\omega_D t) \end{cases} \tag{16}$$

Assume $\left(\begin{bmatrix} u_1 \\ u_1^A \end{bmatrix}, \begin{bmatrix} y_1 \\ y_1^A \end{bmatrix} \right)$ is ASD. Also let the ***Matching Condition:***

$$R(\Gamma_1(x_1)) \subseteq R(B_1^A(x_1)) \tag{17}$$

which says $\exists H_* \ni B_1^A(x_1) H_* = \Gamma_1(x_1)$.

Component #2 will represent all the rest of the evolving system and will be assumed to be strictly dissipative by choice of local controllers:

$$\begin{cases} \dot{x}_2 = A_2(x_2) + \varepsilon B_2(x_2) u_2 \\ y_2 = C_2(x_2) \end{cases} \tag{18}$$

The Components are in Admittance-Impedance form so when they are joined $u_1 = -y_2$ and $u_2 = y_1$.

The Adaptive Key Component Controller with Disturbance Mitigation works through the control input-output ports (u_1^A, y_1^A) of Component #1:

$$\begin{cases} u_1^A = G_e y_1^A + G_D \phi_D \\ \dot{G}_e = -y_1^A (y_1^A)^T \gamma_e ; \gamma_e > 0 \\ \dot{G}_D = -y_1^A (\phi_D)^T \gamma_D ; \gamma_D > 0 \end{cases} \tag{19}$$

This produces $x \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow[t \to \infty]{} 0$ with bounded adaptive gains (G_e, G_D) as the following convergence theorem shows:

Theorem 2: Assume that V_1 and V_2 are positive $\forall x \neq 0$ and radially unbounded, and $(A(x), B(x), C(x))$ are continuous functions of x and $S(x)$, above, is positive $\forall x \neq 0$ and has continuous partial derivatives in x. Furthermore, assume:

1) The conditions of Theo.1 are satisfied; so that $\left(\begin{bmatrix} u_1 \\ u_1^A \end{bmatrix}, \begin{bmatrix} y_1 \\ y_1^A \end{bmatrix}\right)$ is **Almost Strictly Dissipative (ASD)**
2) The Matching Condition: $R(\Gamma_1(x_1)) \subseteq R(B_1^A(x_1))$
3) ϕ_D is bounded (or F has only simple imaginary poles and no right half-plane poles)

Then the adaptive Controller (6) produces

$$x \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow[t\to\infty]{} 0$$

with bounded adaptive gains (G_e, G_D) when Component 1 is joined with Component 2 into an Evolved System and the outputs $y_i = C_i(x_i) \xrightarrow[t\to\infty]{} 0$.

Proof: See Appendix.

It should be noted that the above results might only hold on a neighborhood $N_i(0,r_i) \equiv \{x_i / \|x_i\| < r_i\}$. However, then the stability in Theo. 2 is only locally asymptotic to the origin.

VI. ILLUSTRATIVE EXAMPLE

Example 1, which follows, is a two component linear flexible structure Evolving System. The components of Example 1 are stable when they are unconnected components, but the Evolving System fails to inherit the stability of the components. This example will be used to demonstrate the inheritance and lack of inheritance of almost strict dissipativity in Evolving Systems.

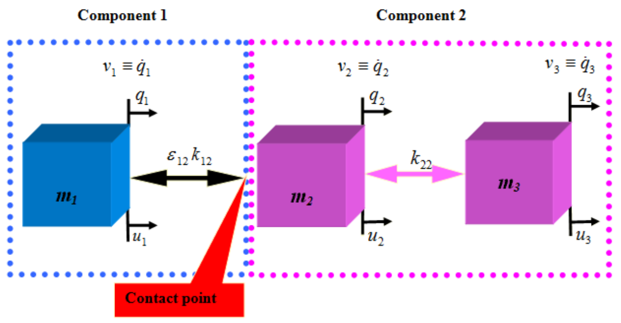


Figure 2. Example 1: A two component flexible structure Evolving System.

The dynamical equations for the components of Example 1 are:

$$\text{component 1:} \begin{cases} m_1\ddot{q}_1 = u_1 - \varepsilon_{12}k_{12}(q_1 - q_2) \\ y_1 = [q_1, \dot{q}_1]^{\mathrm{T}} \end{cases}$$

$$\text{component 2:} \begin{cases} m_2\ddot{q}_2 = u_2 - \varepsilon_{12}k_{12}(q_2 - q_1) \\ -k_{22}(q_2 - q_3) \\ m_3\ddot{q}_3 = u_3 - k_{22}(q_3 - q_2) \\ y_2 = [q_2, \dot{q}_2]^{\mathrm{T}} \\ y_3 = [q_3, \dot{q}_3]^{\mathrm{T}} \end{cases} \quad (20)$$

with $m_1 = 30$, $m_2 = 1$, $m_3 = 1$, $k_{12} = 4$, and $k_{22} = 1$. Example 1 has the following controllers:

$$\begin{cases} u_1 = -(0.9s + 0.1)q_1 \\ u_2 = -\left(\frac{0.1}{s} + 0.2s + 0.5\right)q_2 \\ u_3 = -(0.6s + 1)q_3 \end{cases} \quad (21)$$

When two components join to form an Evolved System, at their point of contact, their velocities are equal and the forces exerted are equal and opposite. If the two components are given by (f_1, v_1) and (f_2, v_2), then the contact dynamics of the Evolved System can be represented by:

$$\begin{cases} f_1 = -f_2 \\ v_1 \equiv \dot{q}_1 = v_2 \equiv \dot{q}_2 \end{cases} \quad (22)$$

This connection can be modeled as the admittance of one component connected in feedback with the impedance of the other component [1]-[3]. When we use this idea of the joining of two components of an Evolving System as the feedback connection of their admittance and impedance, we can apply Theorem 1 from above to determine whether almost strict dissipativty is inherited by the Evolved System.

The subsystem components from Example 1 are stable in closed-loop form when they are unconnected, i.e., $\varepsilon_{12} = 0$. When $\varepsilon_{12} = 1$, the system is fully evolved and it has a closed-loop eigenvalue at 0.17, resulting in an unstable Evolved System.

A Simulink model was created to implement an adaptive key component controller for Example 1 as described in the previous section. Simulations were run in which the connection parameter, ε_{12}, ranged from 0 to 1, allowing the system to go from unconnected components to a fully Evolved System. The key component controller was able to maintain system stability during the entire evolution process when it used the input-output ports on mass 1 of component 1, see Figure 3. When component 1 was the key component, $\left(\overline{A},\overline{B},\overline{C}\right)$ is ASPR.

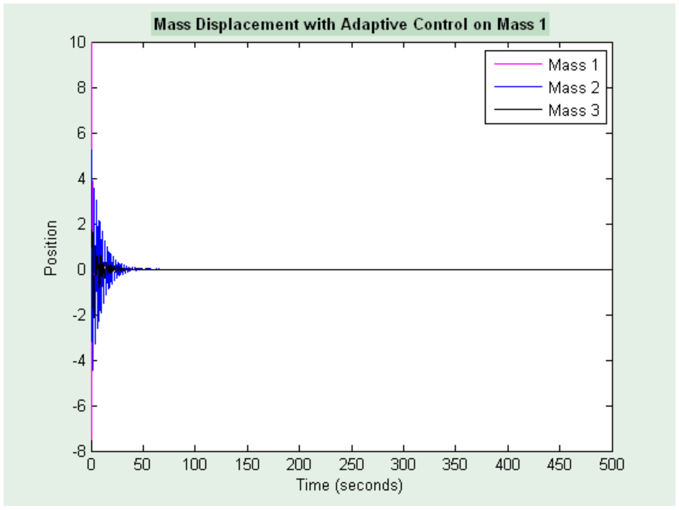


Figure 3. Adaptive key component controller on mass 1.

When the key component controller was located on component 2 and used the input-output ports on mass 3, stability was not maintained, see Figure 4. The adaptive key component controller was not able to restore stability on mass 3 because that system was not ASPR, i.e., it had nonminimum phase zeros at 0.00515±0.2009i.

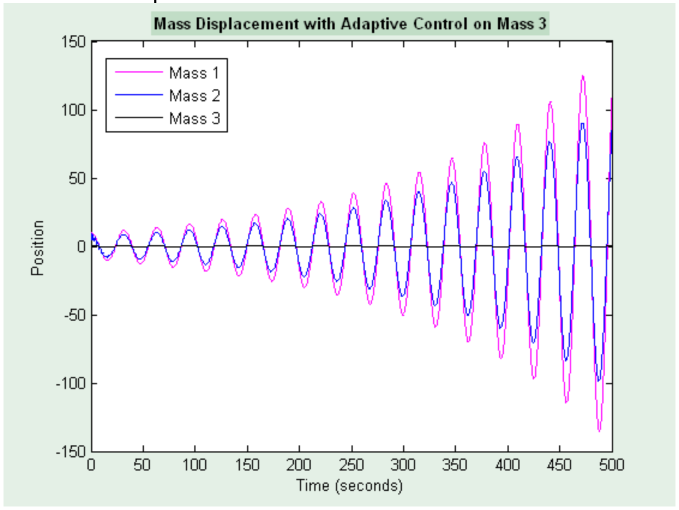


Figure 4. Adaptive key component controller on mass 3.

VII. CONCLUSION

We have presented a result (Theorem 1) describing when an Evolving System will inherit the almost strict dissipativity traits of its subsystem components. An example was given of successful inheritance of almost strict dissipativity and failed inheritance of almost strict dissipativity. This result allows a control system designer to determine a sufficient condition for an Evolving System to use an adaptive key component controller to restore stability. We also presented a convergence result (Theorem 2) for an adaptive key component controller to restore stability during evolution and mitigate persistent disturbances.

REFERENCES

[1] M. J. Balas, M. J., S. A. Frost, and F. Y. Hadaegh, "Evolving Systems: A Theoretical Foundation," *Proceedings AIAA Guidance, Navigation, and Control Conference*, Keystone, CO, 2006.

[2] M. J. Balas and S. A. Frost, "An Introduction to Evolving Systems of Flexible Aerospace Structures," *Proceedings IEEE Aerospace Conference*, Big Sky, MT, 2007.

[3] S. A. Frost and M. J. Balas, "Stability Inheritance and Contact Dynamics of Flexible Structure Evolving Systems", *Proceedings 17th IFAC Symposium on Automatic Control in Aerospace*, Toulouse, France, 2007.

[4] M. J. Balas and S. A. Frost, "Evolving Systems: Inheriting Stability with Evolving Controllers", *Proceedings 47th Israel Annual Conference on Aerospace Sciences*, Tel-Aviv, Israel, 2007.

[5] S. A. Frost and M. J. Balas, "Stabilizing Controllers for Evolving Systems with Application to Flexible Space Structures," *Proceedings AIAA Guidance, Navigation, and Control Conference*, Hilton Head, SC, 2007.

[6] S. A. Frost and M. J. Balas, "Adaptive Key Component Controllers for Evolving Systems," *Proceedings* AIAA *Guidance, Navigation, and Control Conference*, Honolulu, HI, 2008.

[7] R. J. Fuentes and Ml J. Balas, "Direct adaptive rejection of persistent disturbances," *Journal of Mathematical Analysis and Applications*, Vol. 251, No.1, 2000, pp. 28-39.

[8] Susan Frost and Mark Balas, "Evolving Systems and Adaptive Key Component Control", chapter in Aerospace Technologies Advancements, Thawar T. Arif, editor, 2010,ISBN: 978-953-7619-96-1.

[9] CD Johnson, "Theory of Disturbance Accommodating Control," in Control & Dynamic Systems, C.T. Leondes, editor, Vol. 12, 1976.

[10] V.M. Popov, Hyperstability of Control Systems, Springer, Berlin, 1978.

Appendix

Proof of Theorem 1:

Let (u_i^A, y_i^A) be ASD. From (9) and (11),

$\exists G_i^*$ such that

$$\begin{cases} \nabla V_i A_i^C(x_i) \equiv \nabla V_i[A_i(x_i) + B_i^A(x_i) G_i^* C_i^A(x_i)] \\ \qquad \le -S_i(x_i) + \varepsilon_{ij} \nabla V_i A_{ij}(x_i, u_i, u_i^A) \\ \nabla V_i B_i^A(x_i) = C_i^A(x_i)^T \end{cases} \quad \text{(A.1)}$$

If we connect (u_1, y_1) in feedback with (u_2, y_2), then $y_1 = u_2$ and $u_1 = -y_2$ and, use (12) and (13), then we have $\nabla V_1 A_{12}(x_1, u_1, u_1^A) = \nabla V_1 B_1(x_1) u_1 = C_1^T(x_1)[-y_2] = -y_1^T y_2$ and similarly, $\nabla V_2 A_{21}(x_2, u_2, u_2^A) = y_2^T y_1$.

Let $x \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ and, from (12),

$$\begin{cases} \dot{x} = A(x) + B(x)u \\ \quad = \begin{bmatrix} A_1^C(x_1) + \varepsilon_{12} A_{12}(x_2) \\ A_2^C(x_2) + \varepsilon_{21} A_{21}(x_1) \end{bmatrix} \\ \quad + \begin{bmatrix} B_1^A(x_1) & 0 \\ 0 & B_2^A(x_2) \end{bmatrix} \begin{bmatrix} u_1^A \\ u_2^A \end{bmatrix} \\ y = \begin{bmatrix} y_1^A \\ y_2^A \end{bmatrix} = C(x) = \begin{bmatrix} C_1^A(x_1) \\ C_2^A(x_2) \end{bmatrix} \end{cases} \quad \text{(A.2)}$$

with $V = V_1 + V_2$, using (13) and $\varepsilon_{ji} = \varepsilon_{ij} \equiv \varepsilon$ from (3),

$$\begin{aligned} \nabla V A(x) &= [\nabla V_1 \quad \nabla V_2] \begin{bmatrix} A_1^C(x_1) + \varepsilon A_{12}(x_2) \\ A_2^C(x_2) + \varepsilon A_{21}(x_1) \end{bmatrix} \\ &= \nabla V_1 A_1(x_1) + \varepsilon(-y_1^T y_2) + \nabla V_2 A_2(x_2) + \varepsilon(y_2^T y_1) \\ &\le -[S_1(x_1) + S_2(x_2)] + \varepsilon(-y_1^T y_2) + \varepsilon(y_2^T y_1) \\ &= -S(x) \end{aligned}$$

and

$$\begin{aligned} \nabla V B(x) &= [\nabla V_1 \; \nabla V_2] \begin{bmatrix} B_1^A(x_1) & 0 \\ 0 & B_2^A(x_2) \end{bmatrix} \\ &= \begin{bmatrix} C_1^A(x_1) \\ C_2^A(x_2) \end{bmatrix}^T = C^T(x) \end{aligned}$$

Therefore $\left(u_A \equiv \begin{bmatrix} u_1^A \\ u_2^A \end{bmatrix}, y_A \equiv \begin{bmatrix} y_1^A \\ y_2^A \end{bmatrix} \right)$ is ASD with output feedback $\begin{bmatrix} u_1^A \\ u_2^A \end{bmatrix} \equiv \begin{bmatrix} G_1^* & 0 \\ 0 & G_2^* \end{bmatrix} \begin{bmatrix} y_1^A \\ y_2^A \end{bmatrix} + \begin{bmatrix} u_1^{Ar} \\ u_2^{Ar} \end{bmatrix}$ as desired. #

Proof of Theorem 2:

Since the physical connection of Component 1 to Component 2 is equivalent to the feedback connection

$$u_1 = -y_2 \text{ and } u_2 = y_2,$$

By Theo.1 we have that the closed-loop system (u_1^A, y_1^A) below is ASD:

$$\begin{cases} \dot{x}_1 = A_1(x_1) - \varepsilon B_1(x_1) C_2(x_2) + B_1^A(x_1) u_1^A \\ \dot{x}_2 = A_2(x_2) + \varepsilon B_2(x_2) C_1(x_1); 0 \le \varepsilon \le 1 \\ y_1^A = C_1^A(x_1) \end{cases} \quad \text{(A.3)}$$

Rewrite (19),

$$\begin{cases} u_1^A = G_e y_1^A + G_D \phi_D = G_e^* y_1^A + G_D^* \phi_D + \underbrace{\Delta G \eta}_{w}; \\ \Delta G \equiv G - G^* = [\Delta G_e \quad \Delta G_D]; \eta \equiv \begin{bmatrix} y_1^A \\ \phi_D \end{bmatrix} \\ \Delta \dot{G} = \dot{G} = -y_1^A (y_1^A)^T \gamma; \gamma \equiv \begin{bmatrix} \gamma_e & 0 \\ 0 & \gamma_D \end{bmatrix} > 0 \end{cases} \quad \text{(A.4)}$$

Combining (A.3) and (A.4) yields:

$$\begin{cases} \dot{x}_1 = A_1^C(x_1) - \varepsilon B_1(x_1) C_2(x_2) \\ +[B_1^A(x_1) G_D^* + \Gamma_1(x_1) \theta L] \varphi_D + B_1^A(x_1) w \\ \qquad \text{with } w \equiv \Delta G \eta \\ \text{ and } G_D^* \equiv -H_* \theta L \text{ from(19)} \\ \qquad \text{and } A_1^C(x_1) \equiv A_i(x_i) + B_i^A(x_i) G_i^* C_i^A(x_i) \\ \dot{x}_2 = A_2(x_2) + \varepsilon B_2(x_2) C_1(x_1); \ 0 \le \varepsilon \le 1 \\ y_1^A = C_1^A(x_1) \end{cases} \quad \text{(A.5)}$$

Let $V = V_1 + V_2$ and we have:

$$\dot{V} = -S(x) + \langle y_1^A, w \rangle \quad \text{(A.6)}$$

Form $V_G \equiv \frac{1}{2} tr(\Delta G \gamma^{-1} \Delta G^T)$ and obtain from (A.3):

$$\begin{aligned}\dot{V}_G &\equiv tr(\Delta \dot{G} \gamma^{-1} \Delta G^T) \\ &= -tr(y_1^A (y_1^A)^T \Delta G^T) \\ &= -tr(y_1^A (w)^T) \\ &= -\langle y_1^A, w \rangle\end{aligned} \tag{A.7}$$

Define: $V(x, \Delta G) \equiv V(x) + V_G(\Delta G)$ and, from (A.5) and (26), we have:

$$\begin{aligned}\dot{V}(x, \Delta G) &\equiv \dot{V}(x) + \dot{V}_G(\Delta G) \\ &= -S(x) + \langle y_1^A, w \rangle - \langle y_1^A, w \rangle \\ &= -S(x) \le 0\end{aligned} \tag{A.8}$$

This guarantees that all trajectories $(x, \Delta G)$ are bounded. If $\dot{V}(x, \Delta G)$ is uniformly continuous or $\ddot{V}(x, \Delta G)$ is bounded, then Barbalat's Lemma [10] yields:

$$S(x) \xrightarrow[t\to\infty]{} 0,$$

and the positivity and continuity of $S(x)$ imply that

$$x \equiv \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \xrightarrow[t\to\infty]{} 0.$$

Consider

$$\begin{aligned}\ddot{V}(x, \Delta G) &= -\dot{S}(x) \\ &\le \left|\dot{S}(x)\right| \\ &= \left|\frac{\partial S(x)}{\partial x}\dot{x}\right| \\ &\le \left\|\frac{\partial S(x)}{\partial x}\right\| \left\|\dot{x}\right\| \\ &\le \left\|\frac{\partial S(x)}{\partial x}\right\| \left[\left\|A(x)\right\| + \left\|B(x)\right\| \left\|w_1^A\right\|\right] \\ &\le \left\|\frac{\partial S(x)}{\partial x}\right\| \left[\left\|A(x)\right\| + \left\|B(x)\right\| \left\|\Delta G\right\| \left\|C_1^A(x_1)\right\|\right]\end{aligned}$$

which is bounded because $(x, \Delta G)$ is bounded, $S(x)$ has continuous partial derivatives and $(A(x), B(x), C(x))$ are continuous, and a continuous function of bounded $x(t)$ is also bounded in *t*.

So, $y_i = C_i(x_i) \xrightarrow[t\to\infty]{} 0$ because $C_i(x_i)$ is continuous.#

Multi-agent Hierarchical System Based on ElGamal Decryption Algorithm With K+1 Access Levels

Iulia Ştefan, Laura Végh, George Moiş, Stelian Flonta, Szilárd Enyedi, Liviu Miclea
Technical University of Cluj-Napoca, Romania
Iulia.Stefan@aut.utcluj.ro, laura.vegh@gmail.com, George.Mois@aut.utcluj.ro, sflonta@colim.ro,
Szilard.Enyedi@aut.utcluj.ro, Liviu.Miclea@aut.utcluj.ro

Abstract—**In an Internet connected world, security represents a priority. This paper discusses a usage scenario of the ElGamal encryption algorithm with k+1 access levels, by generating a hierarchical tree from the private key algorithm. It underlines the effectiveness in data transfer between different agents as nodes in a hierarchical society. An agent-based system was implemented. This structure was generated using as starting point the algorithm underlined above. Also, the context-awareness authentication is discussed considering the fact that system should provide access to encrypted text.**

Keywords: ElGamal decryption algorithm; hierachical agent model.

I. INTRODUCTION

In an expanding virtual world, where information is stored as binary values and therefore can be copied, security becomes a mandatory aspect. Products which ensure data protection are obtained by increasing development costs. These costs are motivated by the amount of personal computers [2] in use that shows the human interest toward online connectivity, storage, shopping, search, communication, mobility; in fact, toward information interchange, all actions needing protection of private personal data and payments details. In May 2009 [3], Computer Industry Almanac Inc. concluded that a quarter of the worldwide population was using the Internet, almost 1.5 billion users. This situation obliges the software providers to develop and search for improved ways to protect privacy and data. This paper presents a method to use in software agents technology for establishing security levels by key generation.

Section 1 presents current situations in electronic threats trying to underline some of the basic methods to impose security of information. Section 2 describes the usage of the ElGamal algorithm for generating the public and private keys, in order to determine hierarchical access to information. Section 3 presents a prototype of a usage scenario in an agent based hierarchical structure. In Section 4, we discuss the possibility of adding context awareness to a structure, where its objectives need restrictive interaction with the environment.

II. ELECTRONIC THREATS

Mutual authentication between a mobile user and a service provider in [4] represents an attempt to describe a model for such an authentication, key definition, and privacy in access control.

Electronic credential and authentication services are used to verify and link a user to an individual's identity, which will be used within an information system to support the online channel of government service delivery [5].

In GRID applications, the authentication process is based on the Virtual Organization concept. The VO administrator is the one conferring access and establishing some sort of agreements with the Resource (CPU, Network Storage) Providers [10].

In some multimedia applications, authentication establishes the encryption levels [13]. In order to protect the digital media content, the number of trusted parties able to decrypt is reduced to a few.

The need of enhanced security is obvious, when considering the factors representing possible attacks: information the user knows or possesses that can be replicated or stolen. There are different possibilities to reduce the risk of an attack, by implementing a robust authentication process with several credential elements, or using hardware tokens. Several Internet Banking applications, using one-time passwords, impose difficulties to a possible attacker to find the password. Encryption increases the security by adding obstacles in identifying the correct information.

When the situation imposes authentication, there are multiple menaces regarding the authentication event: user, password, sequence of steps exposed to external observers, identity substitution, password guessing, or account modification by intercepting an authenticated session.

There are several ways to reduce the risk exposure: the authentication field's protection against visual interception, reducing the allowed number of login attempts (username, password, biometric credentials), retyping authentication credentials to re-display certain information, no authentication process recording allowed, strict rules when creating credentials to prevent fast guessing or memorizing.

Authentication depends, in fact, on: known information (user, password, number of successive steps in the authentication protocol), hardware devices at hand (token, smart card), and biometric identifiable characteristics (fingerprint, palm print). In the end, the security level relies on the number of security elements implemented.

A way to avoid visual identification of credential granting and the authentication process is automated registrations, key generation and transfer, authentication processes between software agents charged with such functionalities.

A model for such security controls is a hierarchical system, where the upper nodes see all the information designated for the lower ones.

For a software agent involved in message decryption, we considered as contextual information: IP, username, schedule, time and location, all in order to obtain the appropriate private key.

III. ENCRYPTION USAGE

Encryption is a key enabling technology for content security.

With encryption, from an initial document is obtained a new one, with no immediate meaning or readability, by applying a cipher (an algorithm) that usually associates a different symbol to a known linguistic one. To recreate the initial message - to decrypt the message -, the key is necessary, as it represents the information that reveals the connection between the contents.

The ElGamal decryption algorithm with k+1 differential degrees of access rights [16] suggests a tree structure model - the entities claiming the decryption are located in nodes of a tree. Thus, every post can be encrypted by X_i with public key and decrypted by Y_i with private key. There is also the possibility that a group of all posts or messages can be decrypted by other users, using a special private key. A grade 0 user can decrypt all the messages, a grade one user can decrypt a subset of messages that can be decrypted by grade 0 key and so on; a k user can decrypt a single message.

We will now describe the ElGamal decryption algorithm with k+1 differential degrees of access rights EG(k+1)GA.

The algorithm uses the information as a multitude of reduced size messages $M = \{m_{ijk,}\ m_{ijk} \epsilon \{0,1\}, \forall i, j, k \epsilon \mathbb{N}\}$. Let us consider the set I of k pairs of natural values indices. The following notations will be necessary: i… represents $(i_{1,} i_{2,} \ldots, i_k)$ pairs of indices obtained by concatenation and i-k... describes the pairs of indices as $(i_{1,} i_{2,} \ldots, i_{k-j})$ where $j \epsilon \{1,\ldots,k\}$. The information $\{m_{i\ldots} \mid i \in I\}$ is encrypted using a public key by entity X_i and decrypted by Y_i entities using private keys. The intention is that Y^{k-j} k-j decrypts with a private key only the messages included in a partition of the set m_i. Y^0 will be the only one capable to decrypt the entire message.

The algorithm for messages is divided into several steps: generating the tree and creating the node indices, generating the keys, encrypting and decrypting the messages. If necessary, an access level could be eliminated or added.

Defining the tree structure means establishing the users, access levels and the hierarchy: nodes, levels, arcs. The first step is to generate the tree by establishing the indices associated to every node. Every node and the user, associated to a leaf or node, will possess a private key; every leaf has a public and a private one. The indices in a node are generated by pre-ordered tree traversal; each value is given by the child node that was visited. This stage establishes also the level access for every private key.

It is possible to decrypt a message by a private key if and only if there is a chain formed by descendent nodes from the analyzed key to leaf z_i where the message to be decrypted is situated. To generate the private keys, a cyclic group of order q will be chosen, q being a prime number, for which the discrete logarithm problem is difficult; g is its generator.

In case of longer messages, the algorithm could be improved with a symmetric system.

The difficulty in solving the discrete logarithm problem and calculating all the possibilities is given by the number of group G elements. The chosen prime number q is big enough to be represented using 256 bits, and G is cyclic group of order q. Nowadays, the dimension of current keys is equal to 512 and 1024 bits; sometimes, even a larger number is necessary. Another possibility is to use Sophie - Germain prime numbers to prevent external attacks.

From the set $\{1,...,q-1\}$, k distinctive elements will be chosen: $x_1, x_2,..., x_k$. The following functions are chosen $\theta_i: \mathbb{N}^* \to \mathbb{N}^*$, i=1,...,k, function generated by irreducible polynomials.

From mathematical point of view, a private key is a function defined as:

- for the zero level of the tree, the notation for private keys is SK(k+1)G0 and the formula is:

$$f^0 : \underbrace{N^* \times N^* \times .. \times N^*}_{k} \to Z^*_q. \quad (1)$$

$$f^0(n_1, n_2, \ldots, n_k) = x_1^{\Theta 1(n1)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \quad (2)$$

- for the first level of the tree , SK(k+1)G1

$$f_1^1 : \underbrace{N^* \times N^* \times .. \times N^*}_{k-1} \to Z^*_q. \quad (3)$$

$$f_1^1 (n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(n1)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \quad (4)$$

$$f_2^1 : \underbrace{N^* \times N^* \times ... \times N^*}_{k-1} \to Z^*_q. \quad (5)$$

$$f_2^1 (n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(2)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \quad (6)$$

...

$$f_r^1 : \underbrace{N^* \times N^* \times ... \times N^*}_{k-1} \to Z^*_q. \quad (7)$$

$$f_r^1 (n_2, n_3, \ldots, n_k) = x_1^{\Theta 1(r)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)} \quad (8)$$

…

- for the second level of the tree , SK(k+1)G2:

$$f_{11}^2 : \underbrace{N^* \times N^* \times ... \times N^*}_{k-2} \to Z^*_q. \quad (9)$$

$$f_{11}^2(n_3, \ldots, n_k) = x_1^{\Theta 1(1)} x_2^{\Theta 2(2)} \ldots x_k^{\Theta k(nk)} \quad (10)$$

$$f_{12}^2 : \underbrace{N^* \times N^* \times ... \times N^*}_{k-2} \to Z^*_q. \quad (11)$$

$$f_{12}^2 (n_3, \ldots, n_k) = x_1^{\Theta 1(2)} x_2^{\Theta 2(n2)} \ldots x_k^{\Theta k(nk)}$$

…

$$f_{1r}^1 : \underbrace{N^* \times N^* \times ... \times N^*}_{k-2} \to Z^*_q. \quad (12)$$

$$f_{11}^{2} : \underbrace{N^* \times N^* \times \ldots \times N^*}_{k-2} \to Z^*_q. \quad (9)$$

$$f_{1r}^{1}(n_3, \ldots, n_k) = x_1^{\Theta 1(r)} x_2^{\Theta 2(r)} \ldots x_k^{\Theta k(nk)} \quad (13)$$

…

In the end, by the reunion of the **SK(k+1)G$_l$** private keys, the complete set of private keys can be obtained.

$$\bigcup_{l=0}^{k} \mathbf{SK(k+1)Gl = SK(k+1)GA.} \quad (14)$$

A public key to encrypt the message is given by a set of three values tuples as:

P(k+1)K ={(q, g, $h_{i\ldots}$) | i... ϵI }, where $h_{i}\ldots= g^{zi\ldots}$
and the set of private keys SK(k+1)GA becomes the set {$z_{i\ldots}$, f^j, j=0,…, k-1}. Every $Y_{i\ldots}$ user (situated in a node) receives a $z_{i\ldots}$ key and the users situated in lower access levels get keys contained in the set{f^j, j=0,…,k-1}; f^0 is able to decrypt all messages.

To encrypt the message m_i , the node having the public key chooses the elements $X_{i\ldots}$ ϵ{1,…,q-1} and the following equations are obtained:

$$c^1_{i\ldots} = g^{y_i\ldots},\ c^2_{i\ldots} = m_i.\ h^{yi\ldots}_{i\ldots}. \quad (15)$$

The encrypted message is: {$c^1_{i\ldots}$, $c^2_{i\ldots}$}.

The node Y_i uses q and the private key z_i… to decrypt a k level message {$c^1_{i\ldots}$, $c^2_{i\ldots}$}:

$$\frac{c^2_{i\ldots}}{{c^1_{i\ldots}}^{z_{i\ldots}}} = \frac{m_{i\ldots} h^{y_{i\ldots}}_{i\ldots}}{(g^{y_{i\ldots}})^{z_{i\ldots}}} = \frac{m_{i\ldots} h^{y_{i\ldots}}_{i\ldots}}{(g^{y_{i\ldots}})^{z_{i\ldots}}} = \frac{m_{i\ldots} g^{z_{i\ldots} y_{i\ldots}}}{g^{y_{i\ldots} z_{i\ldots}}} = m_{i\ldots} \quad (16)$$

To obtain a decryption on a k-j level, one needs a set of descendent nodes from the key to the leaf z $_{i\ldots}$ of the $m_{i\ldots}$ message.

Starting from the idea of the algorithm presented above, a multi-agent system can be developed where each node in the tree represents an agent.

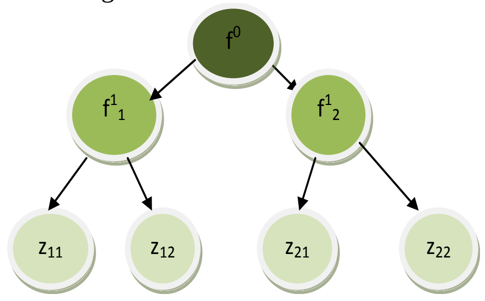


Figure 1. Hierarchical structure

The algorithm assumes the existence of a management entity to generate and share the keys. This entity is also modeled as an agent, without being included in the tree structure. This improves the security, because this entity manages all the keys, both private and public ones.

For example, considering k degrees of access, the tree structure will be as shown in figure 2.

Generating keys is actually creating functions. An initial cyclic group of order q is chosen, q prime number, for which the discrete logarithm problem is difficult; g is its generator. From the set {1, ..., q-1}, k elements and k distinct functions are chosen to be used for private keys calculation. These functions represent irreducible polynomials. To implement these calculations, the Java programming language was chosen.

A function of degree *i* is obtained from an *i-1* grade function, by assigning values to variables in a polynomial component generated in the previous step. For example, for a system with *k+1* degrees of access, the function f^0 has *k* variables, the functions f^1 (on level 1) have *k-1* variables, functions f^2 have *k-2* variables. In the end, the private keys are obtained by assigning values to all variables.

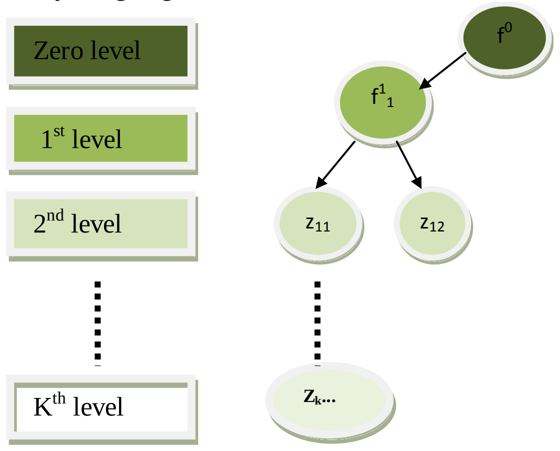


Figure 2. Correlation between level and tree structure

By the method of defining a key, the keys can be constructed one from the other, from level 0 towards level k+1. The reverse is impossible. The Java BigInteger data type is used to represent the key values; it allows storing very large values - an essential requirement for ensuring a high degree of security of the algorithm.

The main objective is to construct a hierarchical structured tree using the private key generation algorithm [17]. This represents one of many ways to build such a system, having as starting point a discrete algorithm problem.

Because the work now is focused on the tree generation, the discussion regarding the decisional Diffie-Hellman assumption and why it holds for chosen group G will be the object of future work, when the system will be improved with the encryption/decryption capabilities. To exemplify, a model for hierarchy implementation is described next, a model implemented in Java Agent DEvelopment Framework(JADE).

The application is based on two packages: one is algPack containing the algorithm implementation classes and the other, the agent implementation classes.

"algPack" is composed of KeyGeneration.java for key generation, Encryption.java for message encryption and decryption.java for decryption.

The second package describes two types of agents. The first is KeyManager; without it, the application does not run. It is in charge of defining the tree structure, starting the agents, generating the keys and transmitting them towards the other agents, the node agents representing the second type of agents..

Initially, online key transfer was considered, but, due to the lack of security in that solution, we consider the possibility to transfer the private keys by offline means, in person.

The tree structure is based on a zero level agent and the other agents placed on inferior hierarchical branches. The zero level agent is characterized by several children, but no parent; the lower levels and the leaf nodes do have parent nodes and children nodes. There is no structural differentiation between leaf node and lower level node because it is important that the tree structure could expand, based on contextual needs.

The KeyManager Agent extracts the necessary data for the tree structure: the number of levels, number of agents and an array describing the number of subordinate agents for every agent. The next phase concerns starting agents and index generation, based on the already mentioned array.

The zero level agent has the index zero. We are assuming it has two children placed on level one, each being described by a single index, 1 and 2. Next, we consider the agent indexed with 1 and we are assuming it has two subordinates (children). Being on the second level as children of first node from level 1, they have two indexes and these are 11 and 12. For every node, its indices are created by copying the parent index and adding at the end its ordinal number until all indices are generated. These indices are necessary for public and private keys and for agent identification in the tree structure. After index generation, the corresponding agents are started. The next phase is represented by key generation and key transfer.

The KeyGeneration class from the algPack package is responsible for key generation. Its constructor receives, as parameters, the array with indices and the number of levels and then generates the public and private keys. These keys are then transferred towards the KeyManager of every agent. Sending the key involves sending a message as ACLMessage.INFORM that a private key will be sent. Because the private key never travels, a trustee will manually install every private key.

Due to the 2:1 expansion of ciphered text over plain text, the consequences are discernible in the necessary volume to store the data and/or the time needed to transfer the encrypted information.

Several BigInteger variables for key storage and manipulation will be set and these variables will be seen as objects in application by using setContentObject method.

Every agent must receive a key. The zero level private key has the highest access level. To receive a message that a key is available from the KeyManager agent, the method "receive" is applied combined with the getContentObject method. If the returned object is not null, a message was transferred and a validation of the message generated by KeyGeneration class is initiated.

In our vision, the agents execute tasks inside a hierarchy. Such tree structures could be identified, for example, in hospitals where the access to private patient data depends on job position or, in software development companies where secrecy represents a management priority in order to maintain the competitors at a safe distance.

In our proposed model, the authority that defines the hierarchy is a supervisor agent. It establishes the private key to send along with the encrypted required information.

The Supervisor Agent will create a hierarchical structure based on stored information about staff and their credentials.

In order to obtain information, field agents running on every network device will send a request message to the supervisor. The supervising authority will send the encrypted information.

The field agent will be able to perceive request information as user/password, date/time, IP, location (inside, outside the buildings) and send it over to the supervisor.

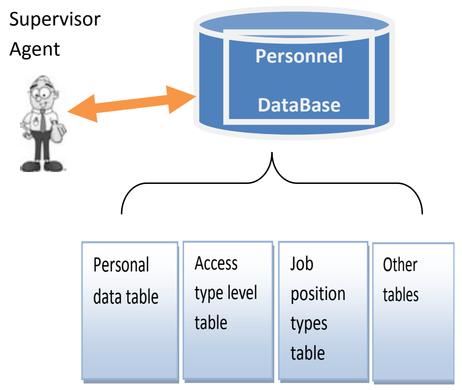


Figure 3. Information resource for Supervisor

Some information, such as a work program or specified scheduled tasks, could be declared inaccessible. For example, why does a lab nurse need to see private data of a patient if he is not scheduled for analysis? The access should be limited to certain time intervals.

IV. CONTEXT AWARENESS

Next, we discuss the possibility to add context-awareness capabilities to the application. This ability begins to become a must in the development of various competitive products, electronic or otherwise, to improve life standards for their owners.

In pervasive computing environments, the humans are continuously surrounded by interconnected devices. Some of these devices are able to access contextual information that can be used as a significant factor in authentication processes, for granting/denying access. User location, date/time, mobile/fixed device used are several examples.

When related to the information content, authentication represents the process to determine, by a pre-established protocol, the identity of the user, user/passwords, user/one-time passwords, using sensors to determine the identity by fingerprint, face characteristics.

„Context includes, but is not limited to the user's location, nearby people and objects, accessible devices, and changes to these objects over time. It may also include lighting, noise level, network connectivity, communication costs, communication bandwidth, and even social situations." [7]. The definition is emphasizing the importance of connectivity and communication.

The user context regarding a computer, mobile device, software driven hardware, "is any information that can be used to characterize the situation of an entity, the user and applications themselves" [11]. In [12], context-aware computing has a major goal: to acquire and fruitfully exploit the information „about context of a device in order to provide services that are appropriate".

But context awareness involves a certain degree of indeterminacy due to its imprecise or incomplete character. For example, not all the sensors send data and, in this case, the retrieved information is not representative for the described situation.

In the case of an agent based structure, designed to offer encryption/decryption capabilities, the service provided should be, at first hand, the authentication. The major problem is the method accurately identifying the user/other service to access the encrypted content. The discussion should be directed toward available means for uniquely identifying a user in our days. Some methods are based on RFID tags, fingerprint sensors, image processing capabilities (retina scan). Other methods could be added to verify different confirmation variables and restrict user access.

If context awareness capabilities are configured for software agents in user identification and authentication processes, the accuracy of contextual information is a binding condition.

In [8], five methods for context reasoning are emphasized starting from their driving factors: the case of past situations, the logic of inference starting from a predicate definition, the ontology as determinations and structure of objective reality, the probability of uncertain contexts and the pre-established system of rules.

By [9], there are four context aware applications based on two criteria: one regarding the presence of automation and the other involving the application objective: to inform or to command something.

In our model, the software agents are responsible for encrypting/decrypting a certain message, based on context data.

To ensure security, multi agent systems use digital signature, proxy servers' certification and hash tables [14].

In [15], the mobile agents can cooperate based on a framework for authentication, using standards already verified in GRID and WEB services.

Further work will be conducted toward an easier way of establishing hierarchy by context awareness capabilities of the application. Based on several pre-established criteria, the application could allow self-validation.

This paper presents an encryption algorithm that allows data access only for an authorized group within a distributed system. This way, sensitive data can be read / modified only by trusted entities in the system. In the decryption algorithm, every node involved in the process is an agent. The decryption key is generated by a higher authority and transferred manually to increase security. The advantage of using such an algorithm to generate the key is the possibility to add nodes without changing all the private keys in the structure.

V. CONCLUSION

The paper represents a starting point in future work related to the architecture of hierarchical structures and possible usage of encryption/decryption algorithms toward security enhancement in information interchange. It discusses the usage of context awareness capabilities in an encryption/decryption hierarchical system. Sensors could provide enough information to uniquely identify a user, but the main issue still remains: the determination if data provided are accurately obtained because the delivered information allows user access to encrypted content and private key.

REFERENCES

[1] http://www.nationmaster.com/graph/med_per_com-media-personal-computers&date=2005, April 29th 2011,

[2] http://www.c-i-a.com/pr012010.htm, April 1th 2011,

[3] http://www.c-i-a.com/pr0509.htm, April 3th 2011,

[4] K. Ren, W. Lou, K. Kim, and R. Deng, A novel privacy preserving authentication and access control scheme for pervasive environments, IEEE Transactions on Vehicular Technology 55 (4) (2006), pp. 1373-1384,

[5] http://www.cio.gov.bc.ca/local/cio/standards/documents/standards/electronic_credential_authentication_standard.pdf, April 28th 2011, pp. 15,

[6] L. Chun-Ta, H. Min-Shiang, and C. Yen-Ping, Further Improvement on a novel privacy preserving authentication and access control scheme for pervasive computing environments, Computer Communications 31(2008), www.elsevier.com/locate/comcom, April 3th 2011, pp. 4255-4258,

[7] K. Ohbyung , S.M. Jong, and K. Seong, Context-aware multi-agent approach to pervasive negotiation support systems, Expert Systems with Applications 31 (2006), pp. 275–285,

[8] Z. Daqiang, G. Minyi, Z. Jingyu, K. Dazhou, and C. Jiannong, Context reasoning using extended evidence theory in pervasive computing environments, Future Generation Computer Systems 26(2010), pp. 207-216,

[9] B. Schilit, N. Adams, and R. Want, Context-Aware Computing Applications, Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications, ISBN:978-0-7695-3451, pp. 85-90,

[10] R. Alfieria, R. Cecchinib, V. Ciaschinic, L. dell'Agnellod, A. Frohnere, K. Lo Renteyf , and F. Spatarog , From gridmap-file to VOMS: managing authorization in a Grid environment, Future Generation Computer Systems 21 (2005), pp. 549–558,

[11] A.K. Dey and G.D. Abowd, Toward a better understanding of context and context awareness, 1999, GVU technical report GIT-GVU-99-22, pp. 3-4,

[12] K. Ohbyung, S. Jong Min, and K.W. Seong, Context-aware multi-agent approach to pervasive negotiation support systems, Expert Systems with Applications 31 (2006), pp. 275–285,

[13] H. Kodikara Arachchi, X. Perramon, S. Dogan, and A.M. Kondoz, Adaptation–aware encryption of scalable H.264/AVCvideo for content security, Signal Processing: Image Communication 24 (2009), link: www.elsevier.com, pp. 468–483,

[14] C. Ou, C.R. Ou, and Y.T. Wang, Security of Mobile Agent-based Web Applications, 2008 IEEE Asia-Pacific Services Computing Conference, 2008, pp. 107-112,

[15] G. Navarro-Arribas and J. Borrell. An XML Standards Based Authorization Framework for Mobile Agents. In Secure Mobile Ad-hoc Networks and Sensors. Springer Verlag, vol. 4075 of Lecture Notes in Computer Science, 2006, pp. 54-66,

[16] S. Flonta, Contributions to the development of models for accessibility and security of information systems, PHD Thesys, 2010, pp. 78-86,

[17] S. Flonta, L. Miclea, and I. Paun, ElGamal with differentiated decryption on K+1 access levels, Applied Computational Intelligence and Informatics, 2009. SACI '09. 5th International Symposium on, Timisoara 2009, pp. 375-380.

Using a Network of Untrusted Computers for Secure Computing

Michal Malý
Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics, Comenius University
Bratislava, Slovak Republic
Email: maly@ii.fmph.uniba.sk

***Abstract*—This paper defines a new problem in distributed computing: How to securely compute one's own problem using a network of untrusted computers? A theoretic solution of the problem is also presented. All secret input data and a secret result are known only by the initiator of the computation. Other computers are used only to carry out the computation using their computer time and memory. The communication can be eavesdropped, and any of untrusted computers can arbitrarily tamper the computation, assuming that no cooperation between untrusted computers occurs. This problem is different from a well-known multiparty distributed computation problem. Here, only one party has all the secret data. The computation is carried out on untrusted computers. An untrusted computer has access to only a small part of secret. Provided no or only a few untrusted computers are cooperating, the secret – input data, and the result of the computation – is not revealed.**

Keywords- cloud computing; security; distributed computing.

I. Introduction

Suppose a company wants to solve a confidential computational task, but they do not own enough computers to carry out the computation on their own computers. They could rent some computer time from another company to carry out the computation on their computers, but they are afraid that the company would misuse the confidential data and the result of the computation.

However, they can rent some computer time from many companies. They believe that those concurring companies would not cooperate together. The computation could be distributed among many untrusted units, assuming that the attack is not synchronized among multiple untrusted parties.

The problem is to execute the computation on untrusted computers so each of the computers has access to only a small part of the computation, i.e., either only to every j-th tape position or to every j-th written symbol for a chosen j. This means, if a confidential document is stored on the tape, an attacker can read it only discontinuously, e.g., only one bit from every ASCII character. This suffices to keep the content confidential. Current methods in distributed computing do not provide any guaranteed protection of this form.

The structure of the paper is as follows. We investigate alternatives which provide at most practical difficulty or provide only protection of stored data. Also related problems in multi-party computation and mobile cryptography are presented. A framework of computers simulating a Turing machine by message passing is presented, first dealing with passive interception and then handling active messages tampering.

II. Alternatives

A. Hiding computation problem

The company could attempt to hide the computation in some way of obfuscating it enough, to prevent anyone from reading the result easily. The company can for instance run a virtual machine inside the provider's computer. If virtualization is done by proprietary software, it can be challenging to analyze all data structures by the potential attacker to figure out what computation is done inside the virtual machine. However, this is only "obfuscation" and should not be considered to be a good practice. A close analysis or some background info can in principle remove the obfuscation layer, uncovering the secrets. The protection should come from a logical impossibility rather than practical or technical difficulty.

B. Hiding data problem

On the other side, if the company's problem would be only storing a vast amount of data, they could simply rent some storage and keep their data in encrypted form there. As long as the act of decrypting is carried on trusted computers, no information leak can occur.

III. Related work

A. Use of distractive units

A method for distributed computation using distractive computational units was patented by Google [2]. The computation is partitioned into computational units and a number of distractive units is generated. All units are forwarded to providers. The results are collected and evaluated in order to obtain the final result. It is supposed, that the distractive units can substantially inhibit the reconstruction of the secret data.

B. *Secure multi-party computation*

A related important problem in cryptography, first introduced in [13], is to enable two or more parties to publicly compute a shared function from secret inputs provided by the parties, guarantying the secrecy of inputs and the correctness of the result. The notorious example are two millionaires, who wish to know, which of them is richer, without revealing the exact wealth. Effective protocols providing protection against active adversaries were proposed by [4] [5] [7] and for multiple parties assuming some level of honesty in [1].

C. *Volunteer computing*

Computer owners readily donate their computer time to interesting projects, such as prime search [6], search for extraterrestrial intelligence [11], and simulations of protein folding [12]. These projects require dealing with an occasional volunteer computers malfunction, intentional fraudulence in order to gain extra credit, or sabotage. The techniques include majority voting and more advanced "spot-checking" [9].

In volunteer computing, each participant has access to the data, which he obtained, and partial result of the computation, which was carried out on his computer. In "public" projects, there is no need to keep it in secret. On the contrary, the result is often shown to the user in order to attract more volunteers, or even a prize is awarded for successful computation.

However, in such conditions, certainly no military computing project could be carried out on volunteer computers.

D. *Mobile cryptography*

In [8], a homomorphic encryption scheme was proposed to enable secure computation on a mobile agent residing on a potentially untrusted hardware. Unfortunately, only encrypted computation of polynomial functions is known.

IV. Problem specification

Let us suppose the company can gain (rent) any reasonable number of computers, each from another party (e.g., in a volunteer project). Let us suppose the parties are not cooperating: Although each untrusted computer is under somebody else's control, nobody has control over two different computers. The goal is to execute a computation on private data, so only the originator of the computation has access to data and to the result of the computation. Each untrusted computer will have access only to a arbitrarily small part of the secret.

A. *Premises*

The network is considered fast enough and the number of network messages used to accomplish the computation is allowed to be linear with the time of computation.

V. Method

The possibility of such computation will be demonstrated using a classical Turing machine model to formalize the computation. The process of computation will rely on standard cryptography mechanisms:

- To ensure confidentiality of transferred messages, the originator generates a digital signature for himself and for each host to encrypt and sign messages.
- To prevent tracing of messages, the mechanism of anonymous routing called "onion routing" [3] and implemented in TOR [10] is used. The originator and each hosts connects to the TOR network, and only originator knows the role of each host.

The process will be accomplished in two steps. First, I will suppose nobody is actively tampering the computation, only passive eavesdropping is allowed. Assuming this I prove that only an arbitrary small part of computation is revealed to each untrusted side.

The second step involves the mechanism of majority voting, which at each step of the computation, that a malicious message will be detected and the attacker will be revealed.

VI. Description

The framework uses a network of computers passing messages between them. The network as a whole will do the computation. To make the computation secret, the computation is split to small parts. No computer in the network can gain a large amount of information at one time. The order of computers participating in the computation is devised so as no host can intercept successive parts of the computation.

To describe the computation, we will use the Turing machine. Every movement of the head will be a small, atomic part of the computation. One part consists from messages which are passed between respective computers. After executing this part, the computation moves one step ahead, and another sequence of messages is passed between another set of computers.

Theorem 1: Let us have a Turing machine $A = (K, \Sigma, \Gamma, \delta, q_0, F)$, where $|K| = k$, and t be the time, i.e., the t-th step of the computation. Let $n > 1$ be a natural number. Let w be the input word. Then it is possible to execute the computation of the machine A on the word w using a network of $k{*}n{+}n$ independent untrusted computers, so that if nobody is tampering the computation:

1) if $A(w)$ stops after T steps, then the execution in the networks stops after T iterations, gives the same result, and the number of messages in the network is $5 * T + c * k * n$ where c is some constant.
2) i-th computer can only have access to $(n * j + i)$-th position on the tape during the whole computation for any j and $0 \leq i < n$ and $(n + r * k + s)$-th computer can only have access to the symbols read or written to

the tape in $(n*j+r)$-th step of network computation where $0 \leq r < n$, $0 \leq s < k$ and any j.

Proof: The setup is as follows: The host computers are split into two groups. First group of n computers will represent the tape; second group of $k*n$ computers represents the head of the machine. Computers in first group will form a cycle, so i-th computer represents every $(n*j+i)$-th position on the tape for any j and for $0 \leq i < n$.

Computers in the second group will represent the head, i.e., the finite-state automaton. To accomplish this, let us take k computers, associate each of k states of the automaton to one of k computers, and program the computer so it knows the values of delta function for its associated state. Now we have this distributed automaton, which is able to act as a head of the machine. But if the delta function is constructed so the automaton remains in one state for a long time, one computer would be able to intercept a long part of the computation.

To avoid this, we make n copies of this automaton, and rewire the outputs of delta function to point to the respective state, but not in the same copy as the current computer (state), but in the following copy of the automaton. The computers in the last copy of the automaton will point to the computers in the first copy, so the copies form a cycle (in some sense). Now we connect the head to the tape. We describe one transaction, which represents one movement of the head. The transaction consists of the following messages (see Fig. 1):

1) The originator obtains addresses of the network computers.
2) The originator generates number identification for each computer (according to the numbering in this proof).
3) The originator generates digital keys for each of computers and uploads them.
4) Each of the computers joins TOR network and communicate its address within TOR network to the originator.
5) The originator discloses number identification, public key, and TOR address of each computer to every computer.
6) The originator uploads an computing protocol application on each computer.
7) The originator fills out input data on tape.
8) The originator starts the execution.
9) The network computation iterates, each iteration is composed of an atomic transaction between 3 computers, see below.
10) When the computation stops in one of final states, the computer representing the state sends message to the initiator.
11) The initiator is now able to retrieve the content on the tape.

The transaction (see Figure 1) runs between 3 computers: The computer representing the current state of the automaton, the computer representing current tape position and the computer representing the new tape position.

Suppose we are in the t-th step of computation and Turing machine state is the s-th state ($0 \leq s < k$) and tape is in position p. Let r, r' be remainders of $t, t+1$ divided by n.

First, the current-state computer ($(n+r*k+s)$-th) asks for the symbol on the tape. It receives an answer from the current-tape-position computer ($p \text{ modulo } n$). Let this symbol be x. According to the delta function it decides what is the new state (s'), what symbol is to be written on the tape (y), and where to move the head (direction $d \in \{-1, 0, 1\}$), i.e., $\delta(q_s, x) = (q_{s'}, y, d)$. Therefore it sends a message with new symbol y and the movement direction d to the current-tape-position computer, which will remember the new symbol, and will send a message to the computer which represents the new tape position ($(p+d) \text{ modulo } n$). This message will authorize the current state to announce the new state ($(n+r'*k+s')$-th computer) to the new-tape-position computer. Which computer will be the new-state computer is derived from the delta function and from the number of copy of the automaton, in which belongs the current-state computer. The new-state computer is chosen to represent the new state s', but in the next copy r' of the automaton. This new-state computer is thus authorized to read the symbol from the new position. ■

VII. Byzantine computing

How can we assure nobody is actively modifying the computation? We could use more computers, to check them each other. In principle we run the same computation on more computers and detect if the outcome (the messages sent) differs (majority voting).

Suppose at most one computer is violating the protocol, and we want to detect this violation and report the violator. If more computers will be cheating, the result is undefined.

The setup is as follows. Every computer is tripled (create a copy of it, assign the copies the same role, only the public key and TOR address is different). Messages previously sent to the computer are now sent parallel to each of its copies. Now each computer waits until it receives messages from all copies of its predecessors. It compares the received messages. If they are different, it reports to the originator and sends received messages. Messages are signed, so the report cannot be faked. According to this report from all three copies, the central computer can determine who violated the protocol.

A timeout can be defined to enable detection of sabotage by not sending messages. After receiving the first message, the computer waits until all messages arrive. A missing message is reported after the time-out to the originator.

The number of computers allowed to violate the protocol (v) can be extended in a similar manner using $2v + 1$

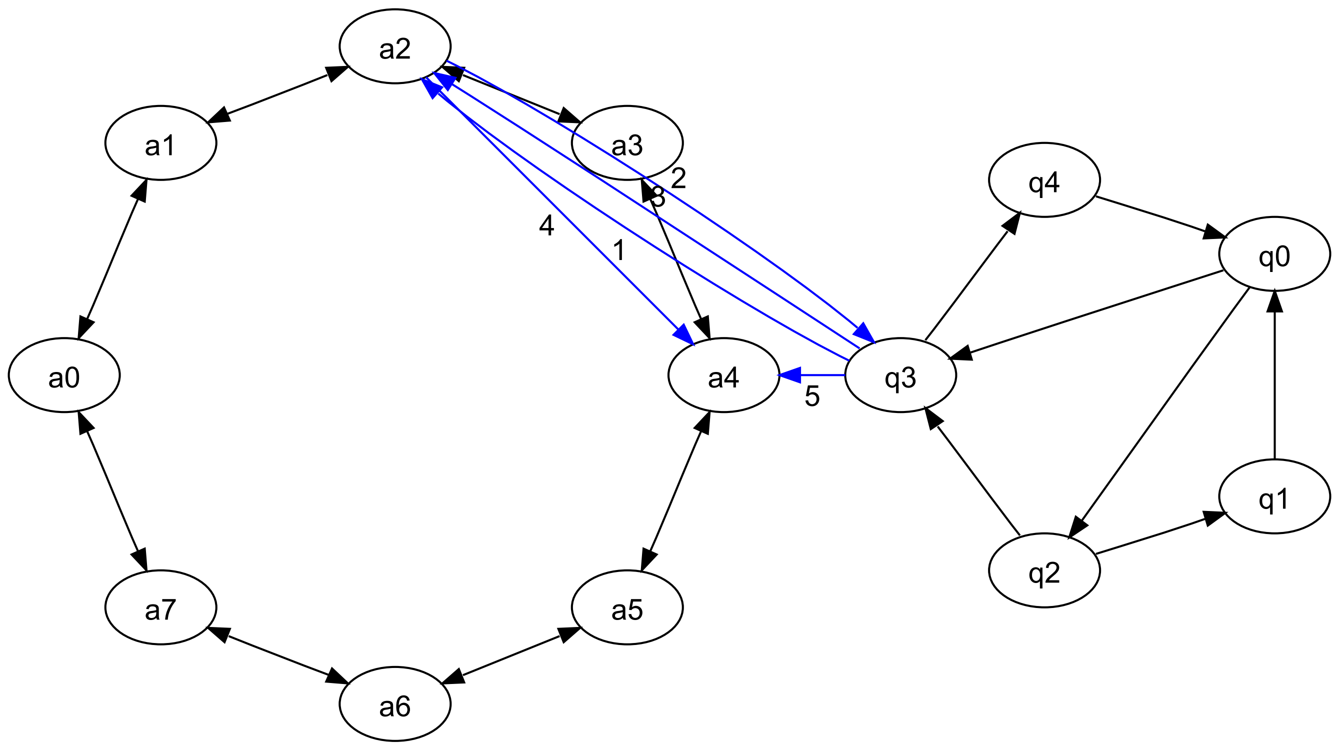


Figure 1. Example of an atomic transaction. The states a_y represent the tape, the states q_x represent the automaton. For simplicity we show only one copy of automaton. Black arrows are the possible transitions between states, blue arrows are messages. Messages: #1 What is on the tape? #2 Symbol 'x'. #3 Write 'y' and move head left. #4 we are moving left and q3 is the old state. #5 The new state is q4 (but from the next copy of automaton).

independent copies.

VIII. Applications

Perpetually growing business of computer outsourcing or modern trends such as web applications can release us from owning and maintaining hardware. The presented method allows us to not resign on security issues. The method can be used even in intra-company applications: The company could have a number of not-so-trusted administrators. The system based on this computational model could save the company from the possibility of disclosing the secrets to them. A military, or other confidential research could be realized on a number of volunteer computers.

IX. Conclusion and Future Work

We have presented a method to run a computation on a number of computers, where no computer has a complete access to the computation and the result of the computation is not revealed. In fact, the part of the computation revealed to a single computer can be made arbitrarily small.

The classical Turing machine model is not flexible enough to reflect current hardware possibilities, and was used to demonstrate the theoretical possibility of the problem solution. Another, more refined model based on RAM (random access machine), using individual computers as separate registers, will be investigated in future work.

Even using the Turing machine model, some bad-designed programs can reveal the whole secret, when the head is accessing the tape too many times. This can be possibly solved by re-designing the program. The number of messages is huge, and similarly the number of used computers is relatively high. However, the method demonstrates the theoretical possibility of trusted computing on untrusted computers.

References

[1] David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 11–19, New York, NY, USA, 1988. ACM.

[2] S.N. Gerard. Distributed computation in untrusted computing environments using distractive computational units, February 9 2010. US Patent 7,661,137.

[3] D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *Information Hiding*, pages 137–150. Springer, 1996.

[4] Stanisaw Jarecki and Vitaly Shmatikov. Efficient two-party secure computation on committed inputs. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 97–114. Springer Berlin / Heidelberg, 2007.

[5] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 52–78. Springer Berlin / Heidelberg, 2007.

[6] Mersenne Research, Inc. Great Internet Mersenne Prime Search. Available at: http://www.mersenne.org, 1996–2011. Last access: May-11-2011.

[7] Jesper Nielsen and Claudio Orlandi. Lego for two-party secure computation. In Omer Reingold, editor, *Theory of Cryptography*, volume 5444 of *Lecture Notes in Computer Science*, pages 368–386. Springer Berlin / Heidelberg, 2009.

[8] T. Sander and C. Tschudin. Protecting mobile agents against malicious hosts. *Mobile agents and security*, pages 44–60, 1998.

[9] L.F.G. Sarmenta. Sabotage-tolerance mechanisms for volunteer computing systems* 1. *Future Generation Computer Systems*, 18(4):561–572, 2002.

[10] The Tor Project, Inc. Tor: anonymity online. Available at: https://www.torproject.org, 2002–2011. Last access: May-11-2011.

[11] University of California. Seti@home. Available at: http://setiathome.berkeley.edu, 2011. Last access: May-11-2011.

[12] Vijay Pande and Stanford University. Folding@home. Available at: http://folding.stanford.edu/, 2000–2011. Last access: May-11-2011.

[13] Andrew C. Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 160 –164, nov. 1982.

Reduced Order Modeling of Linear MIMO Systems Using Particle Swarm Optimization

Dia Abu-Al-Nadi
Dept. Electrical Engineering
University of Jordan
Amman, Jordan
dnadi@ju.edu.jo

Othman MK Alsmadi
Dept. Electrical Engineering
University of Jordan
Amman, Jordan
Othman_mk@yahoo.com

Zaer S Abo-Hammour
Dept. Mechatronics Engineering
University of Jordan
Amman, Jordan
zaer_hr@yahoo.com

Abstract—**In this work, a model order reduction (MOR) technique for a linear multivariable system is proposed using the combined advantage of retaining the dominant poles and the error minimization using the particle swarm optimization. The state space matrices of the reduced order system are chosen such that the dominant eigenvalues of the full order system are unchanged. The other system parameters are chosen using the particle swarm optimization with objective function to minimize the mean squared errors between the outputs of the full order system and the outputs of the reduced order model when the inputs are unit step. The proposed algorithm has been applied successfully, a 10th order Multiple-Input_Multiple-Output (MIMO) linear model for a practical power system was reduced to a 4th order and an 8th order Single-Input-Single-Output (SISO) system was reduced to a 2nd order.**

Keywords-Model Order Reduction;MIMO Systems;Particle Swarm Optimization.

I. Introduction

Many physical systems are translated into mathematical model via higher order differential equations. It is usually recommended to reduce the order of this model while keeping the dominant behavior of the original system. This will help to better understanding of the physical system, reduce computational complexity, reduce hardware complexity and simplify the controller design.

Different techniques for order reduction of linear continuous MIMO system in time domain as well as in frequency domain are available in the literature [1-4]. For model order reduction, there are different scenarios that can be performed. One scenario obtains reduced models that are completely new and not related to the original models in terms of their critical frequencies of either SISO or MIMO systems. On the other hand, another scenario obtains reduced models that preserve the original system important properties, such as dominant frequencies of either SISO or MIMO systems. It is to be noted that the later scenario is more preferable, if possible, due to its meaningful physical interpretation in obtaining similar models and due to minimum changes in the original systems [5].

The MOR problem has been investigated in literature extensively. Willcox and Perarie [6] proposed an algorithm to reduce the model order using the proper orthogonal decomposition (POD) analysis of the primal and dual systems, low-rank, reduced-range approximations to the controllability and observability gramians. Fujimoto and Scherpen [7] proposed a singular perturbation type balanced realization and model reduction for discrete non-linear dynamical systems based on Hankel singular value analysis, which preserves the related controllability and observability properties. Heydari and Pedram [3] proposed a spectrally weighted balanced truncation technique for tightly coupled integrated circuit (IC) interconnects, when the interconnected circuit parameters change because of statistical variations in the manufacturing process. Rabiei and Pedram [8] proposed a method that uses the truncated balanced realization technique as well as the Schur decomposition to develop an efficient numerical method for the order reduction of linear time invariant (LTI) systems. Gugercin et al. [9] proposed an iterative rational Krylov approach (IRKA) for optimal H_2 model reduction. Their approach is concerned with SISO-type systems only and is based on minimizing the H_2-norm. This minimization leads to a non-convex problem that can get stuck at local minima and hence optimality will not be achieved [10]. Recently, Parmar et al. [4] proposed a reduction method with pole centroid retaining in the reduced model. Their method deals with SISO systems only. In addition, it works for real poles only. Genetic algorithm (GA)-based MOR, on the contrary, has received some of the researchers' attention as well. Recently, Panda et al. [11] employed a particle swarm optimization technique to obtain a reduced-order model of SISO large-scale linear systems. Their technique is based on integral square error (ISE). Vishwakarma and Prasad [12] proposed a mixed method for reducing the order of large-scale linear systems. They have synthesized the denominator of the reduced-order transfer function (TF) using modified pole clustering, whereas the coefficients of the numerator elements are computed using GA. Parmar et al. [13] presented a technique for MOR using GA for SISO linear time systems. They have focused on obtaining a reduced-order model that maintains stability and retains the steady-state value.

In this work, the particle swarm optimization (PSO) is utilized for MOR of MIMO systems. The rest of the paper is organized as follows: Section II is the statement of the problem. In section III, the PSO algorithm is stated; section IV is designated for results and discussion, and finally, conclusions are presented in section V.

II. PROBLEM STATEMENT

MOR is investigated both for MIMO and SISO systems, for the MIMO systems, the state space representation was adopted while for the SISO systems, the transfer function model is used.

A. MOR for MIMO systems

Consider the following nth order LTI system:

$$\dot{x}_f(t) = A_f x(t) + B_f u(t) \tag{1}$$

$$y_f(t) = C_f x(t) + D_f u(t) \tag{2}$$

where $x_f \in \Re^n$ is the state vector, $u \in \Re^p$, and $y_f \in \Re^m$ are the input and output vectors, respectively. The matrices A_f, D_f, C_f, and D_f are the full order system matrices with their appropriate dimensions. Let the eigenvalues of the above full order system be given as: $-\lambda_1 < -\lambda_2 < \cdots < -\lambda_n$.

On the other hand, consider the reduced order LTI system with order r:

$$\dot{x}_r(t) = A_r x_r(t) + B_r u(t) \tag{3}$$

$$y_r(t) = C_r x_r(t) + D_r u(t) \tag{4}$$

where $x_r \in \Re^r$ is the state vector of the reduced order system, $u \in \Re^p$, and $y_r \in \Re^m$ are the input and output vectors,respectively. The matrices A_r, B_r, C_r, and D_r are the reduced order system matrices with their appropriate dimensions. The eigenvalues of reduced order system are chosen to be the dominant eigenvalues of the full order system given as: $-\lambda_1 < -\lambda_2 < \cdots < -\lambda_r$.

The A_r matrix is chosen to be a diagonal matrix with the dominant eigenvalues are assigned as the diagonal elements. The elements of other matrices are chosen by the PSO.

B. MOR for SISO Systems

Consider an nth order SISO LTI system with the following transfer function:

$$G(s) = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_{n-1} s^{n-1}}{b_0 + b_1 s + b_2 s^2 + \cdots + b_n s^n} \tag{5}$$

with the eigenvalues of the system to be $-\lambda_1 < -\lambda_2 < \cdots < -\lambda_n$.

Let the reduced order model be

$$G_r(s) = \frac{\alpha_0 + \alpha_1 s + \alpha_2 s^2 + \cdots + \alpha_{r-1} s^{r-1}}{\beta_0 + \beta_1 s + \beta_2 s^2 + \cdots + \beta_r s^r} \tag{6}$$

The coefficients of the denominator of $G_r(s)$ are chosen such that the eignenvalues of the low order system are the dominant roots of the full order system as follows: $-\lambda_1 < -\lambda_2 < \cdots < -\lambda_r$, while the coefficients of the numerator are chosen by the PSO algorithm.

III. THE PSO ALGORITHM

The PSO is a multiple-agent optimization algorithm developed by Kennedy and Eberhart [14] in 1995. The major advantage of the PSO over other stochastic optimization methods is its simplicity. The standard PSO is implemented by assuming a swarm of particles (called trial solutions). Each particle moves in the solution space by improving its position according to suitable updating equations. This is performed on the basis of information on each particle's previous best performance and the best previous performance of its neighbors (global best). The updating equations for the PSO are sequentially applied at each individual. Unlike other stochastic algorithms, the PSO is based upon the cooperation among the trial-solutions and not on their competition. In order to describe the steps of the PSO algorithm, we will define the given parameters and the necessary specifications. Hence, two parts can be classified:

A. Definitions and parameters setting:

- Set the full order system parameters.
- Set appropriate level step inputs to the system.
- Simulate the outputs, yf, of the full order system with a suitable sampling time.
- Choose a suitable order of the reduced order system based on the dominant eigenvalues.
- Set the PSO parameters:
- The size of the particle, P.
  - The number of particles in the swarm, *M*.

- o The counter of iteration (I = 1) and the maximum number of iterations, L_{max}.

- Definition of the solution space: A reasonable range for the parameters should be chosen. This requires specifications of the minimum and maximum values for each parameter.

- Definition of a fitness function: This step is the link between the optimization algorithm and the physical problem in hand. A good fitness function that is well representative of the parameters is crucial in the PSO algorithm. In this work, the fitness function is defined by the weighted-mean-squared error

$$WMSE = \frac{1}{N}\sum_{k=1}^{m}\sum_{i=1}^{N} w_k [y_f(k,i) - y_r(k,i)]^2 \qquad (7)$$

where N is the number of samples, m is the number of outputs, w_k is a weight used to emphasize the kth error, $y_f(k,i)$ is the ith sample of the kth output of full order system and $y_r(k,i)$ is the ith sample of the kth output of reduced order system.

In this paper, the fitness function used in the PSO algorithm is the minimization of WMSE

$$fitness = \min(WMSE) \qquad (8)$$

B. The main steps of the PSO algorithm.

Step 1- Initialization:

The PSO starts by randomly initializing the position matrix, X, the velocity matrix, V, and the personal best matrix, P, of each particle in the swarm such that

$$\mathbf{X} = \begin{bmatrix} \mathrm{x}_1 \\ \vdots \\ \mathrm{x}_i \\ \vdots \\ \mathrm{x}_M \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1P} \\ \vdots & \vdots & \ddots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{iP} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MP} \end{bmatrix} \qquad (9)$$

and the velocity matrix is

$$\mathbf{V} = \begin{bmatrix} \mathrm{v}_1 \\ \vdots \\ \mathrm{v}_i \\ \vdots \\ \mathrm{v}_M \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1P} \\ \vdots & \vdots & \ddots & \vdots \\ v_{i1} & v_{i2} & \cdots & v_{iP} \\ \vdots & \vdots & \ddots & \vdots \\ v_{M1} & v_{M2} & \cdots & v_{MP} \end{bmatrix} \qquad (10)$$

The personal best position can be defined by the matrix

$$\mathbf{P} = \begin{bmatrix} \mathrm{pbest}_1 \\ \vdots \\ \mathrm{pbest}_i \\ \vdots \\ \mathrm{pbest}_M \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1P} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i1} & p_{i2} & \cdots & p_{iP} \\ \vdots & \vdots & \ddots & \vdots \\ p_{M1} & p_{M2} & \cdots & p_{MP} \end{bmatrix} \qquad (11)$$

The global best solution gbest is the row of personal best matrix, **P**, with the best fitness function given as

$$gbest = \min(fitness(\mathrm{pbest}_i)) = [g_1 \quad g_2 \quad \cdots \quad g_P] \qquad (12)$$

In most cases, the initial position is the only location encountered by each particle at the start of the algorithm. Hence, it will be regarded as the particle's respective personal best.

Step 2- Particle Updating.

For each iteration, the particles will be moved into the solution space. The algorithm will act on each particle such that each particle will move in a direction to improve its fitness function. The following steps summarize the action encountered on each particle in the swarm:

a) *Update the Particle's velocity.* The particle's velocity will be updated according to three vector elements: the first is the relative location to its corresponding pbesti; the second is its relative location to gbest; and the third vector is a scaled factor of the old velocity. For each particle, the velocity update is

$$\mathbf{v}_i^{t+1} = w^{t+1}\mathbf{v}_i^{t} + c_1\eta_1(\mathbf{pbest}_i^{t} - \mathbf{x}_i^{t}) + c_2\eta_2(\mathbf{gbest} - \mathbf{x}_i^{t}) \qquad (13)$$

The superscript t+1 and t refer to the time index of the next and the current iterations. η_1 and η_2 are two uniformly random numbers in the interval [0,1]. A good choice for c1 and c2 are both 2.0. The parameter wt is a number called the inertial weight which is a scaling factor of the previous velocity of the particles. It has been demonstrated that PSO algorithms converges faster if w is chosen to be linearly damped with iterations [14]. A good choice to start with is w1=0.9 at the first iteration and linearly decreases to wLmax=0.4 with the last iteration.

b) *Movement Updating of the particles:*

Once the velocity of each particle is determined, the position will be updated

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^{t} + \Delta t\, \boldsymbol{v}_i^{t} \qquad (14)$$

For simplicity, Δt is chosen to be unity.

c) Evaluate the fitness function for the new position and compare it with the fitness function of the pbest, if *fitness*($\boldsymbol{x}_i$) < *fitness* (***pbest***$_i$) then ***pbest***$_i$ = $\boldsymbol{x}_i$

d) Compare the fitness function of the new position with the fitness function of *gbest*

if *fitness*($\boldsymbol{x}_i$) < *fitness* (***gbest***) then ***gbest*** = $\boldsymbol{x}_i$

e) Repeat (a), (b), (c), and (d) for the whole M particles.

Step 3- Check if maximum iteration reached or a specified termination criteria is satisfied. Then, the solution is gbest. Otherwise, update w and go to the next iteration.

IV. RESULTS AND DISCUSSION

To demonstrate the proposed method of the PSO model reduction, we will consider two dynamical examples. The first one is 2-input 2-output, 10th order power system represented with its state space full order system [15]. The second example is a single input single output 8th order transfer function [16].

Example 1

Consider the following 2-input, 2-output, 10th order power system with the following state space model:

$$A_f = \begin{bmatrix} -0.5517 & 0 & -0.3091 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1695 \\ -0.0410 & 0 & -0.0350 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 314.1593 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 9.5540 & 0 & -0.8660 & -20 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0.0421 & -0.0328 \\ -0.1962 & 10.8696 & -0.1672 & 0 & 0 & -10.8696 & 0 & 0 & 0 & 0 \\ -0.9386 & 51.9849 & -0.7999 & 0 & 0 & -41.1153 & -10.8696 & 0 & 0 & 0 \\ -0.9386 & 51.9849 & -0.7999 & 0 & 0 & -41.1153 & -10.86.96 & -0.1 & 0 & 0 \\ 0 & 0 & 0 & -1000 & -1000 & 0 & 0 & 1000 & -20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0526 & -0.8211 \end{bmatrix}$$

$$B_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0.0926 & 0 & 0 & 0 & 0.4428 & 2.1179 & 2.1179 & 0 & 0 \end{bmatrix}^T$$

$$C_f = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.4777 & 0 & -0.0433 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$D_f = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The eigenvalues of the full order system are:

{-18.9311 ± 2.0250i, -12.1968, -9.6484, -0.2394 ± 3.2350i

-0.8972 ± 1.3560i, -2.1313, -0.1001}

The following eigenvalues were chosen to be in the reduced order model

{-0.2394 ± 3.2350i, -0.8972 ± 1.3560i}

Hence, the reduced order state space model will be

$$A_r = \begin{bmatrix} -0.2394 & -3.2350 & 0 & 0 \\ 3.2350 & -0.2394 & 0 & 0 \\ 0 & 0 & -0.8972 & -1.3560 \\ 0 & 0 & 1.3560 & -0.8972 \end{bmatrix}$$

$$B_r = \begin{bmatrix} x_1 & x_5 \\ x_2 & x_6 \\ x_3 & x_7 \\ x_4 & x_8 \end{bmatrix} \quad C_r = \begin{bmatrix} x_9 & x_{10} & x_{11} & x_{12} \\ x_{13} & x_{14} & x_{15} & x_{16} \end{bmatrix} \quad D_r = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The coefficients of the B_r and the C_r matrices are evaluated using the PSO algorithm as follows:

$$B_r = \begin{bmatrix} -2.4213 & -1.4544 \\ -3.5196 & 1.0311 \\ 2.5894 & -2.1142 \\ -.9881 & 0.7407 \end{bmatrix}$$ and

$$C_r = \begin{bmatrix} 1.3483 & -2.5905 & 2.4427 & -2.3762 \\ -0.1990 & -0.1990 & 0.1311 & 0.4203 \end{bmatrix}$$

Simulating both models (the full and the reduced) to a step input are shown in Figure 1.

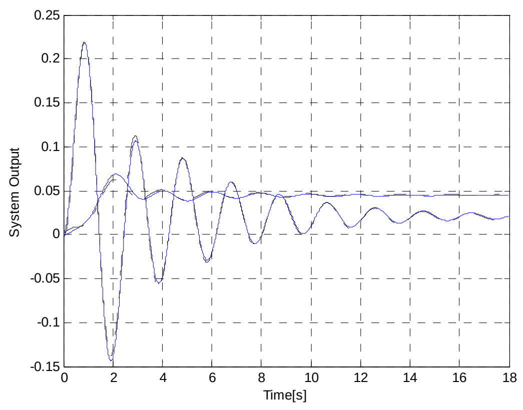


Fig.1: The two outputs of the system for both the full and the low order models caused by a step input for Example 1.(The full order outputs; solid, the reduced order outputs; dashed)

Example 2

Consider the 8th order transfer function

$$G(s) = \frac{18s^7 + 514s^6 + 5982s^5 + 36380s^4 + 122664s^3 + 222088s^2 + 185760s + 40320}{s^8 + 36s^7 + 546s^6 + 4536s^5 + 22449s^4 + 67284s^3 + 118124s^2 + 109584s + 40320}$$

The eigenvalues of the full order system are { -1, -2, -3, -4, -5, -6, -7, -8}. The reduced order model is designed to have the following transfer function (TF):

$$G_r(s) = \frac{x_1 s + x_2}{s^2 + x_3 s + x_4}$$

Coefficients of the numerator and the denominator are evaluated using the PSO. Hence,

the reduced order system was obtained as

$$G_r(s) = \frac{17.0989s + 5.0742}{s^2 + 6.9722s + 5.1514}$$

with poles {-6.1321 and -0.8401}. Again, simulating the full and the reduced order models to a step input produced the output shown in Figure 2.

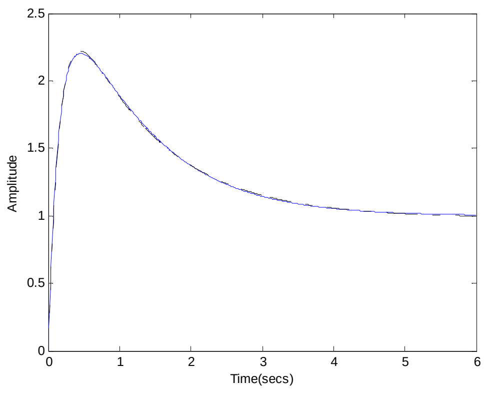


Fig.2: The output of the system for both the full and the reduced order models caused by a step input for Example 2. (The full order output; solid, the reduced order output; dashed)

V. Conclusion and Future Work

Model order reduction has been implemented by using the PSO algorithm. The behavior of the original system was preserved in the WMSE sense. The dominant poles of the full order system were kept unchanged in the reduced order model. The retaining of the dominant poles will guarantee that the overall behavior of the reduced order system will be almost the same as the original system. Based on the results obtained by the illustrative examples, it is concluded that the proposed method achieved satisfactory results. As for future work, implementation of this technique on actual physical systems will take place. Systems with very large dimensions will be considered to explore the powerfulness of the method. Also, comparison of the PSO algorithm with other evolutionary optimization techniques will be investigated.

References

[1] M.G. Safonov and R.Y. Chiang, A Schur Method for Balanced-Truncation Model Reduction, IEEE Trans on Automatic Control 34 (1989), pp. 729 –733.

[2] E. Fridman, Robust Sampled-Data H_∞ Control of Linear Singularly Perturbed Systems, IEEE Transactions on Automatic Control 51 (2006), pp. 470-475.

[3] P. Heydari and M. Pedram, Model-Order Reduction Using Variational Balanced Truncation with Spectral Shaping, IEEE Transactions on Circuits and Systems I 53 (2006), pp. 879-891.

[4] G. Parmar, S. Mukherjee, and R. Prasad, System reduction using factor division algorithm and eigen spectrum analysis, Applied Mathematical Modeling 31 (2007), pp. 2542–2552.

[5] T. Reis and T. Stykel, Balanced truncation model reduction of second-order systems, Mathematical and Computer Modeling of Dynamical Systems 14 (2008,), pp. 391-406.

[6] K. Willcox and J. Perarie, Balanced Model Reduction via the proper orthogonal decomposition, AIAA. J. ,40 , 2002, pp. 2323-2330.

[7] K. Fujimoto and J.M.A. Scharpen, Balancing and Model Reduction for Discrete-Time Nonlinear System based on Hankel Singular Value Analysis, Proceedings of the MTNS 2004, Leuven, Belgium, July 2004, pp. 343-347.

[8] P. Rabiei and M. Pedram, Model-Order Reduction of Large Circuitsusing Balanced Truncation, Proceedings of the IEEE ASP-DAC, Wanchai, Hong Kong, Jan. 1999, pp. 237-240.

[9] S. Gugercin, C. Athanasios, A.C. Antoulas, and C. Beattie, A Rational Krylov Iteration for Optimal H2 Model Reduction, Proceedings of the 17th Int. Symposium on Mathematical Theory on Networks and Systems, Kyoto, Japan, 2006, pp. 1665-1667.

[10] G. Obinata and B.D.O. Anderson, Model Reduction for Control System Design, Springer-Verlag, London, 2001.

[11] S. Panda, J.S. Yadav, N.P. Padidar, and C. Ardil, Evolutionary Techniques for Model Order Reduction of Large Scale Linear Systems, Int. J. Applied Sci. Eng. Technol.5 (2009) ,pp.22-28.

[12] G. Parmar M.K. Pandey, and V. Kumar, System Order Reduction Using GA for Unit Impulse Input and a Comparative Study Using ISE and IRE, International Conf. on Advances in Computing, Communications and Control, Mumbai, India , Jan. 2009, pp.23-24.

[13] C.B. Vishwakarma and R. Prasad, MIMO System Reduction Using Modified Pole Clustering and Genetic Algorithm, Modeling and Simulation in Engineering 2009(ID 540895) 2009, pp. 1-5.

[14] J. Kennedy and R.C. Eberhart, " Particle Swarm Optimization", in Proc. IEEE Conf. Neural Networks IV, Piscataway, NJ , 1995.

[15] G. Parmar, S. Mukherjee, and R. Prasad, Reduced order Modeling of linear MIMO Systems Using Genetic Algorithm, Int. J. simul. Model 6 (2007) 3, pp. 173-184.

[16] S. Mukherjee, M. Satakshi, and R.C. Mittal, Model order reduction using response matching technique, J. Franklin Inst. 342 (2005), pp. 503–519.

Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow

Xavier Dutreilh†, Sergey Kirgizov*, Olga Melekhova*, Jacques Malenfant*, Nicolas Rivierre† and Isis Truck‡
*Université Pierre et Marie Curie – Paris 6 CNRS, UMR 7606 LIP6, 4 place Jussieu, Paris, 75005, France
Email: {Olga.Melekhova, Jacques.Malenfant}@lip6.fr
†Orange Labs, 38-40 rue du Général Leclerc, Issy-les-Moulineaux, 92130, France
Email: {xavier.dutreilh, nicolas.rivierre}@orange-ftgroup.com
‡LIASD – EA 4383, Université Paris 8, 2 rue de la Liberté, Saint-Denis Cedex, 93526, France
Email: truck@ai.univ-paris8.fr

***Abstract*—Dynamic and appropriate resource dimensioning is a crucial issue in cloud computing. As applications go more and more 24/7, online policies must be sought to balance performance with the cost of allocated virtual machines. Most industrial approaches to date use *ad hoc* manual policies, such as threshold-based ones. Providing good thresholds proved to be tricky and hard to automatize to fit every application requirement. Research is being done to apply automatic decision-making approaches, such as reinforcement learning. Yet, they face a lot of problems to go to the field: having good policies in the early phases of learning, time for the learning to converge to an optimal policy and coping with changes in the application performance behavior over time. In this paper, we propose to deal with these problems using appropriate initialization for the early stages as well as convergence speedups applied throughout the learning phases and we present our first experimental results for these. We also introduce a performance model change detection on which we are currently working to complete the learning process management. Even though some of these proposals were known in the reinforcement learning field, the key contribution of this paper is to integrate them in a real cloud controller and to program them as an automated workflow.**

***Keywords*-Cloud computing; virtual machine allocation; reinforcement learning; autonomic computing.**

I. Introduction

Dimensioning resources to applications appropriately is a crucial issue in cloud computing. As applications go more and more 24/7, online policies must be sought to balance performance offered to clients with the cost of virtual machines (VM) for the service provider by making their allocation following closely the workload. Most industrial approaches to date use *ad hoc* manually determined policies, such as threshold-based ones where a low threshold on performance triggers more allocation while a high one triggers a reduction in the number of allocated VMs. Providing good thresholds proved to be tricky and hard to automatize to fit every application requirement [1].

Research is being done to apply automatic decision-making approaches, such as reinforcement learning (RL) [2]. These approaches are particularly well-suited to cloud computing as they don't require the *a priori* knowledge of the application performance model, but rather learn it as the application runs. Yet, RL faces a lot of problems to go to the field [3][4], such as: having good policies in the early phases of learning, time for the learning to converge to an optimal policy and coping with changes in the application performance behavior over time. In this paper, we propose to deal with these problems using appropriate initialization for the early stages, convergence speedups applied throughout the learning phases and performance model change detection. Even though some of these proposals were known in the RL field, the key contribution of this paper is to integrate them in a real cloud controller and to program them as an automated workflow.

We present our first results towards this automated learning management workflow. Section II introduces the resource allocation problem for cloud computing. Section III presents the formulation of the problem in the Q-learning framework, as we have modeled it for a private cloud deployed at Orange Labs. Section IV then presents the core contribution of the paper, the implementation workflow meant to bring RL to real cloud computing infrastructures. Section V then compares to the related work and the conclusion follows. Throughout the paper, experimental results are shown to back up the proposals.

II. Resource allocation in clouds

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [5]. This model promotes availability and is composed of three delivery models: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). IaaS designates the provision of IT and network resources, such as processing, storage and bandwidth as well as management software. PaaS designates the deployment of applications created using particular programming languages and tools supported by a provider onto his own cloud infrastructure.

SaaS designates the use of applications running on a cloud infrastructure. Companies can use clouds either to run data-processing applications (from development tools like continuous integration suites to business tools as video transcoders), transaction-processing software (including social networks and e-commerce websites) or event-processing systems (as fraud detection tools in the financial market). The cloud is appealing to them because of its ability to reduce capital expenditures and increase return on investment since the traditional model where physical hardware and software resources were bought and amortized in the long term is no more.

Many web applications (like social networks) face workload and system changes that affect their performance during their lifetime. First, they have to cope with large fluctuating loads. In predictable situations (like the promotion of a new feature), resources can be provisioned in advance through the use of proven techniques like capacity planning. However, for unplanned spikes (due to slashdotting for instance) and unplannable events (e.g., a rise in popularity after a natural disaster), auto-scaling appears to be the way to automatically adjust resources allocated to applications based on their needs at any given time. In addition, auto-scaling seems to be a lot more justified when it comes to handle changes in applications (following software improvements) and the cloud platform itself (e.g., hardware and software upgrades). The core features of auto-scaling are a pool of available resources that can be pulled or released on-demand and a control loop to monitor the system and decide in real time whether it needs to grow or shrink. Auto-scaling is offered in PaaS environments by providers like Google App Engine and Heroku but applications developed specifically for these platforms are tied to them. In that way, IaaS appears more flexible since users are given free access to virtualized hardware, relying on providers like Amazon [6] and Rackspace or open-source projects like OpenNebula and OpenStack to instantiate VMs. IaaS issues however are the lack of widely adopted standards, although initiatives, such as DMTF and OGF OCCI, are active toward the definition of virtualization formats and IaaS APIs; and automation since developers must build the machinery, or use third party tools, such as RightScale and Claudia . This paper focuses primarily on resource allocation policies that could be used in IaaS and PaaS management layers to perform auto-scaling.

Resource allocation and policies

Fostered by autonomic computing concepts, allocating resources to applications in clouds has been the subject of several works in recent years. As a decision-making problem, the allocation of VMs to an application consists in regularly observing the workload w (in request per second), the current number of allocated VMs u and the current performance p as the average waiting time of requests in seconds and from that, decides to allocate more VMs or deallocate some, in order to maintain the performance p as close as possible to a target performance P given by the SLA of the application while minimizing the costs for the service provider.

Two types of policies have drawn attention in recent works:

1) Threshold-based policies, where upper and lower bounds on the performance trigger adaptations, and where some amount of resources are allocated or deallocated (typically one VM at a time);
2) Sequential decision policies based on Markovian decision processes (MDP) models and computed using, for example, reinforcement learning.

Threshold-based policies are very popular among on-the-field cloud managers. The simplicity and intuitive nature of these policies make them very appealing. However, with the growing complexity of applications, setting thresholds is a per-application task and can be very tricky, especially when it comes to find corrective actions for all possible states [1][7] and deal with performance model changes. As an alternative to manual threshold-based policies, modeling the system as an MDP allows computing policies that can take into account the inertia of the system, such as fixed costs to allocate or deallocate VMs, which favors retaining the same number of VMs when the variation in the workload does not last enough time to amortize these fixed costs. Such compromises are the cornerstones of sequential decision making.

III. The basic RL problem

We now introduce the formulation of the resource allocation problem for clouds as an MDP and the corresponding Q-learning resolution approach implemented in our VirtRL controller [1].

A. Resource allocation as an MDP

With long lasting executions, applications become more and more subject to changes, affecting their end-user performance. To cope with such changes in a decent amount of time and minimize SLA violations, our controller takes resource allocation decisions regularly. As such and as the decisions themselves influence each others over time, the decision-making is modeled as an MDP [8]. Coarsely speaking, an MDP involves a decision agent that repeatedly observes the current state s of the controlled system, takes a decision a among the ones allowed in that state and then observes a transition to a new state s' and a reward r that will drive its decisions. As their name indicates, MDPs are stochastic. Hence, the new state and sometimes also the reward observed from the transition obey probability distributions that characterize the behavior of the underlying controlled system.

The MDP that models our approach to the VM allocation problem is defined as $\mathcal{M} = \langle S, A, T, R, \beta \rangle$ where:

- $S = \{(w, u, p) \mid 0 \leq w \leq W_{max} \wedge 0 \leq u \leq U_{max} \wedge 0 \leq p \leq P_{max}\}$ is the state of the MDP where:
  - $w \in \mathbb{N}$ is the workload in number of requests per second, bounded by $W_{max} = 40$;
  - $u \in \mathbb{N}$ is the current number of homogeneous VMs allocated to the application, bounded by $U_{max} = 10$;
  - $p \in \mathbb{R}$ is the performance expressed as the average response time to requests in seconds, bounded by a value P_{max} chosen from experimental observations.

- $A = \{a \in \mathbb{Z} \mid A_{min} \leq a \leq A_{max}\}$ is the action set which consists in adding, maintaining or reducing the number of homogeneous VMs allocated to the application. The actions have been bounded between $A_{min} = -1$ and $A_{max} = 10$ in our experimental setup;
- $T : S \times A \times S \rightarrow [0, 1]$ is the probability distribution $P(s'|s, a)$ of a transition to new state s' given that the system is in state s and action a is chosen;
- $R : S \times A \rightarrow \mathbb{R}$ is the cost function expressing the expected reward when the system is in state s and action a is taken. When stochastic, it can be expressed as $R : S \times A \times \mathbb{R} \rightarrow [0, 1]$, the probability distribution $P(r|s, a)$ of observing a reward r when the system is in state s and action a is taken;
- $\beta, 0 < \beta < 1$ is a discount factor used to finitely evaluate the overall expected reward for an infinite sequence of decisions. The value $\beta = 0.45$ has been used throughout our experiments.

When the functions T and R can be determined prior to the execution of the controlled system, traditional dynamic programming (DP) algorithms, such as value iteration [2][8] can be applied to find an optimal policy. Value iteration solves the following equation expressing the expected total reward over infinite horizon (for the deterministic reward case):

$$V^*(s) = \max_a \left[R(s, a) + \beta \int_{s' \in S} T(s, a, s')V^*(s') \right] \quad (1)$$

and then the optimal allocation policy is given by:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \left[R(s, a) + \beta \int_{s' \in S} T(s, a, s')V^*(s') \right] \quad (2)$$

The value iteration algorithm does this by successive approximations until a predefined error bound ϵ is reached:

```
$(\forall s \in S),$ initialize $V_0(s)$
$t := 0$
**loop**
    $t := t + 1$
    **foreach** $s \in S$
        **foreach** $a \in A$
            $Q_t(s, a) := R(s, a) + \beta \int_{s' \in S} T(s, a, s')V_{t-1}(s)$
    $\pi_t(s) :=$ argmax$_a$ $Q_t(s, a)$
    $V_t(s) := Q_t(s, \pi_t(s))$
**until** $\sup_s |V_t(s) - V_{t-1}(s)| < \epsilon$
**return** $\pi_t$
```

B. Resolution through Q-learning

The advantage of traditional DP algorithms is that policies are computed offline. The decision-making at runtime then simply amounts to applying the precomputed policy π^* to the sequence of observed states to provide the corresponding actions. However, T and R are often very difficult to estimate. This can require lengthy experimentation and measurement processes upon the actual controlled system and it must be redone each time a modification to the system may change the probability distributions of its transitions or rewards.

To address these limitations, reinforcement learning has been proposed to learn these as the controlled system operates and as the controller is making decisions to learn from experience. Among the different reinforcement learning approaches [2], the Q-learning is based on the equation for $Q(s, a)$ derived from the value function (see equation 1) and appearing in the inner loop of the value iteration algorithm. It turns out that the function $Q(s, a)$, or Q-function, is easy to learn from experience. Given a controlled system, the learning agent repeatedly observes the current state s, takes an action a and then a transition occurs and it observes the new state s' and the reward r. From these observations, it can update its estimation of the Q-function for state s and action a with:

$$Q[s, a] := (1 - \alpha)Q[s, a] + \alpha \left(r + \beta \max_a Q[s', a'] \right) \quad (3)$$

where α is the rate of learning, balancing the weight of what has already been learned with the weight of the new observation. Throughout our experiments, we have used the value $\alpha = 0.8$. The basic Q-learning algorithm is then [2]:

```
$(\forall s \in S)(\forall a \in A(s)),$ initialize $Q(s, a)$
$s :=$ the initial observed state
**loop**
    Choose $a \in A(s)$ according to a policy derived from $Q$
    Take action $a$ and observe next state $s'$ and reward $r$
    $Q[s, a] := (1 - \alpha)Q[s, a] + \alpha (r + \beta \max_a Q[s', a'])$
    $s := s'$
**end loop**
**return** $\pi(s) = \text{argmax}_a Q(s, a)$
```

C. Experimental results

In order to experiment the Q-learning, we have defined a reward function as follows. Given a state $s = (w, u, p)$, an action a, the next state $s' = (w', u', p')$ and a target performance P_{SLA} (notice that $u' = u + a$), $CO(a)$ represents the cost of acquiring and renting the VMs, while $PE(s')$ captures the penalties imposed when the target performance is violated:

$$R(s', a) = CO(a) + PE(s')$$

$$CO(a) = \begin{cases} c_i \times a & \text{if } a > 0 \\ 0 & \text{else} \end{cases} + c_f \times u' \times \Delta t$$

$$PE(s') = \frac{p_c}{3600} \times \Delta t \times \begin{cases} \left(1 + \frac{p' - P_{SLA}}{P_{SLA}}\right) & \text{if } p' > P_{SLA} \\ 0 & \text{else} \end{cases}$$

where:

- $c_r = 0.095$ (US\$ per hour) is the rental cost of VMs per unit of time (it corresponds to the price of a standard on-demand VM on Amazon EC2 [6]);
- $c_i = \frac{1}{60} c_r$ is the initial one-shot cost of getting a new VM when allocated. It is fixed as a fraction of the rental cost for an hour;
- Δt is the length of the time interval between decisions (40 seconds in our experiments);
- $p_c = 10$ (US\$ per hour) is the penalty for the application per unit of time for not providing the level of performance

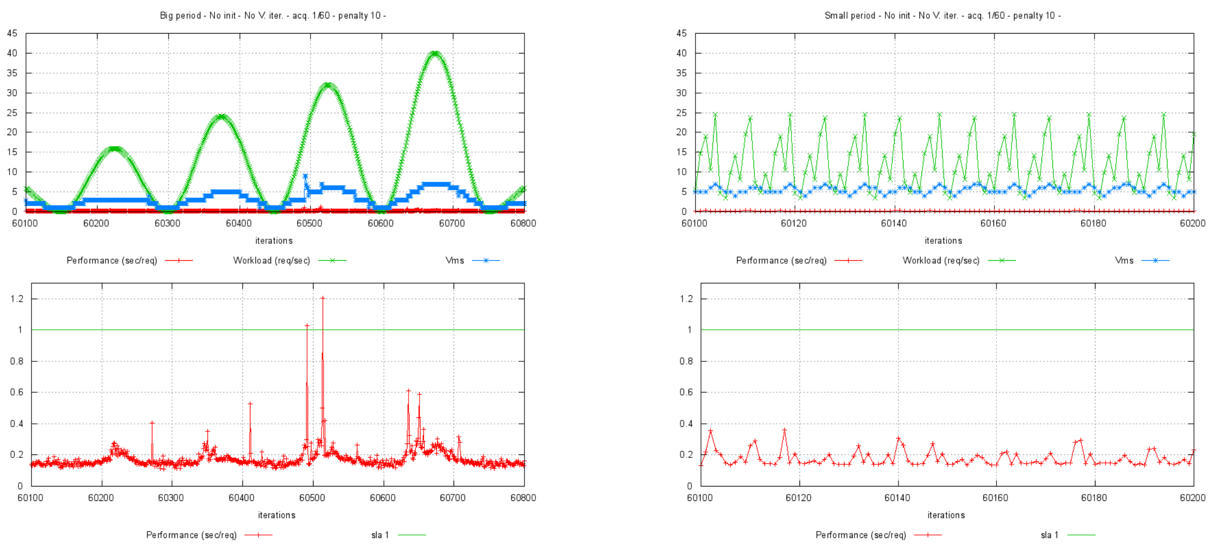


Fig. 1. Comparison of the policies obtained at convergence for two different workload patterns. On the left, a sinusoidal with a dynamic oscillation slope on a long period is used while, on the right, one with a shorter period and a random noise is applied. Beware that the scales are not the same for both cases.

agreed in the SLA. Beware that such a value has direct influence on the aggressiveness and the conservativeness of resource allocation strategies captured by the Q-learning.

This reward function has been used to simulate a cloud executing Olio [9] and the VirtRL decision agent. In this setup, we assume that all VMs used by Olio share a common size which is 1 compute unit (equivalent to 2.66 GHz guaranteed), 256 MB of memory and 20 GB of storage. Figure 1 shows the policies that have been obtained at convergence given two different workload patterns. The first pattern is a long-period sinusoidal function that was used as a baseline for the learning algorithm as its characteristics were easy to learn. The second is a short-period sinusoidal function with random noise, meant to represent the characteristics of real workloads more accurately. Even though these workloads are meant to mimic the cyclic variations of many real-world application workloads, their period has been forced shorter than real ones to put more stress on the learning process.

As we can see from the plots, in both cases, the Q-learning algorithm has converged to an allocation policy that follows the workload with a similar period. The bottom plots give the corresponding performance of the application, the green line giving the target performance while the red line gives the actual one. Besides, to date, our experiments have shown that the time spent in computations and the memory space used to store the representation of the functions are negligible in the cloud computing context.

IV. VirtRL reinforcement learning workflow

Besides the basic learning algorithm, the VirtRL workflow introduces three new activities discussed below:

- Initialization of the Q-function;
- Convergence speedup phases at regular intervals of observations;
- Performance model change detection.

A. Initialization of the Q-learning

The Q-learning algorithm presented above involves an unspecified initialization step for the Q-function. In theory, provided that the mathematical conditions are observed [10], convergence is guaranteed. Pragmatically, however, the distance between the initial Q-function and the one at convergence has two major impacts on the learning process: decisions made during the early phases and time to converge.

In order to apply the control as the learning process is being done, a policy must be followed from which decisions will be chosen and taken on the controlled system. Defining such a policy can be complicated and it also must allow for some exploration of the different possible actions in order for the learning to get the outcome of these actions to discover which is the best one for each state. In the cloud computing context, without prior information about the application, only a standard policy can be applied, such as an ϵ-greedy policy [2], which is a kind of stochastic policy defined as:

$$\pi(s) = \begin{cases} \text{argmax}_a\, Q(s,a) & \text{with probability } 1-\epsilon \\ \text{choose } a \in A(s) \text{ randomly} & \text{with probability } \epsilon \end{cases}$$

For such a policy to give good results, $Q(s,a)$ must be a good approximation of the optimal Q-function at convergence. Until each state has been visited enough times for the learning to update the Q-function to a value approaching the optimal, the current Q-function provides no better information to choose the action than what provides the initial Q-function itself. Hence, for an ϵ-greedy policy to give good results during the first phase of the learning, a good initial Q-function must be found.

As the above value iteration algorithm shows, there is a tight relationship between the value function $V(s)$ and the Q-function. For a given policy π, we have:

$$Q^{\pi}(s,\pi(s)) = R(s,\pi(s)) + \beta \int_{s' \in S} T(s,\pi(s),s')V^{\pi}(s') \quad (4)$$

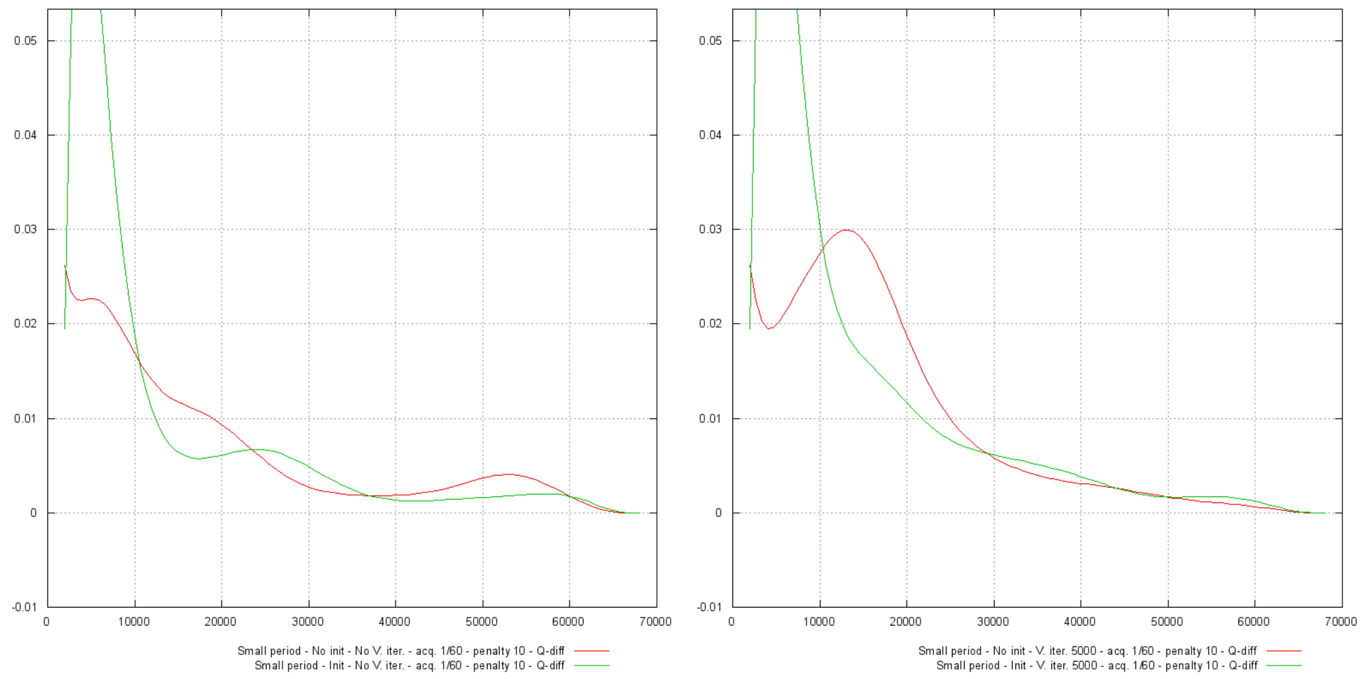


Fig. 2. The two plots show the average difference between two successive Q-functions as learning observations are processed. On the left, no convergence speedup is applied, while on the right, convergence speedups are applied every 5.000 observations. Green curves show the case with initialization, while red ones show learning without initialization. Convergence speedups accentuate the differences between successive Q-functions during the early phases of learning, but diminish them in the middle phase (before 30.000 observations). After 30.000 observations, speedups make little difference.

while the value function itself can be computed for a given policy using the following equation:

$$V^{\pi}(s) = R(s, \pi(s)) + \beta \int_{s' \in S} T(s, \pi(s), s') V^{\pi}(s') \qquad (5)$$

But, of course, we need definitions for the functions R and T, as well as a policy π. Our approach to initialization considers approximate functions $\overline{R}$ and $\overline{T}$, and applies a greedy policy (always choose the best possible action) to find an approximate value function $\overline{V}^{\pi}$, which in turn is put into equation 4 to give an initial Q-function $\overline{Q}$.

Figures 2 and 3 show the result of applying such an initialization, along with convergence speedups to which we return in the next subsection. The approximate reward function that we have chosen is the above reward function truncated of its penalty term (that is much more difficult to estimate). The approximate transition function retains the determinism in the new number of VMs, but considers the next workload and the next performance values as normal random variables centered on the current workload and performance values respectively. The no-initialization case in fact takes $Q \equiv 0$ as the initial Q-function. In Figure 2, we can see that the initialized case exhibits larger average differences at first, but then converges faster to lower differences during a second phase until the overall convergence becomes the same after 30.000 observations approximately. In Figure 3 that is explained in section IV-B, we can see that, after beginning the execution with 10 VMs, the decisions make the allocation come to a less costly number of VMs much faster, though the performance is more volatile. The fact that our current estimated reward function completely neglects the penalties for bad performance, explains this behavior. A bit more guidance than a simple ϵ-greedy policy during this first phase or a more precise estimated reward function, quite simply overcomes these negative effects.

B. Convergence speedups

As presented above, Q-learning learns very progressively the Q-function, updating it only for the visited states and only when they are visited. Compared to the value iteration algorithm, the Q-learning updates the Q-function only for the visited state at each observation, while the value iteration updates the V-function for all states at each iteration. Is it possible to speed up the convergence of Q-learning by using ideas coming from the value iteration algorithm? This is the basic idea behind model-based Q-learning [2][11], which recognizes that there is more to learn from the observations $\langle s, a, r, s' \rangle$ than just from the value of $Q(s, a)$.

Model-based Q-learning uses these observations to estimate, in the statistical sense, the functions R and T. Consider the following measures:

- $C[s, a]$ is the number of times the state s has been observed and the action a taken;
- $T_C[s, a, s']$ is the number of times a transition from state s and action a to the state s' has been observed;
- $R_C[s, a]$ is the sum of the rewards that has been obtained when the state s has been observed and action a taken.

Hence, for each observation $\langle s, a, r, s' \rangle$, these counters are updated as:

$$T_C[s,a,s'] \;:=\; T_C[s,a,s'] + 1 \qquad (6)$$
$$R_C[s,a] \;:=\; R_C[s,a] + r \qquad (7)$$
$$C[s,a] \;:=\; C[s,a] + 1 \qquad (8)$$

Given these statistics, estimators $\bar{T}$ and $\bar{R}$ of the functions T and R are computed as:

$$\bar{T}(s,a,s') = \frac{T_C[s,a,s']}{C[s,a]} \qquad (9)$$
$$\bar{R}(s,a) = \frac{R_C[s,a]}{C[s,a]} \qquad (10)$$

With these estimators, it becomes possible to compute an estimator $\bar{V}^*$ of the optimal V-function V^* and use this estimator to update the current Q-function. This approach to convergence speedups is the most aggressive compared to other partial sweeps of the state space, such as eligibility traces and sample backups model-based techniques [2].

Figure 3 compares the decisions made during the learning process given that convergence speedups are applied or not. For these experiments, we present results for speedups made at each 5,000 observations. At the top, during the early phases of learning, the differences are due to the initialization, as explained in the previous subsection, as no convergence speedup has been performed yet. In the middle, we see that when convergence speedups have been made during the first period of learning, the decisions follow more closely the workload and the performance is more stable. After 30,000 observations, there is no longer a noticeable difference between the two. Convergence speedups therefore help to have a better policy earlier in the learning process.

C. Model changes detection

With long-lasting application executions, it is less and less possible to ignore that the behavior of applications will change over time, a key assumption when applying MDPs to model them. Indeed, applications can be updated with patches or major software releases that change their performance. Moreover, the workload can also change, for example with a different mix of the possible requests hence also changing its average performance. Reinforcement learning, if the learning is pursued during the whole lifetime of the application, can adapt the policy to relatively smooth changes in its behavior [2]. However, the learning will not be able to react to major changes in the behavioral model promptly, especially if the rate of random actions used for learning is decreased over time, as suggested to ease the convergence.

Our proposal is to use again the information gathered with the model-based reinforcement learning approach to do statistical testing upon the observations used to estimate R and T, between older and newer observations to decide whether a model change has occurred or not. As the form of the probability distributions is not known, non-parametric testing shall be used. And because the two samplings are not independent, appropriate tests such as Wilcoxon signed rank test [12] shall be used. We conjecture that such tests can detect major changes in the behavior of the application and that the more subtle changes undetectable with them can be dealt with by using continuous learning all over the application execution. Experiments to back up this claim are currently undertaken.

V. Related work

Several works apply resource allocation techniques in cloud computing. We concentrate here on the ones that take a disciplined rather than an *ad hoc* approach to the problem.

Among the different works on threshold-based policies, Lim *et al.* propose proportional thresholding to adapt policy parameters at runtime [13]. It consists in modifying the range of thresholds in order to trigger more frequent decisions when necessary. This approach adapts very well to fast changing conditions and is directly integrable into automated agents with stability mechanisms. Although the aforementioned paper gives answers to the latency instability, it still lacks of an adaptation of the power of the decisions to fluctuating workloads.

Xu *et al.* propose a two-level controller [14]. A first level applies fuzzy logic techniques to learn the relationship between workload, resources and performance. It then controls the resource allocation by deciding at a regular interval the level of resources needed by each application, thanks to the fuzzy rules previously learned. The second level applies a simply knapsack technique to allocate the resources to the different applications. Though interesting to learn the behavior of applications, this approach limits itself to homogeneous applications by computing only one set of rules and does not really care about stability. Rao *et al.* with their VCONF [4] apply reinforcement learning but in the context of neural networks, in a way that parallels the work of Xu *et al.* Their neural network represents the relationship between workload, resources and performance used to perform vertical scaling.

Tesauro *et al.* explore the application of reinforcement learning in a sequential decision process [3]. The paper presents two novel ideas: the use of a predetermined policy for the initial period of the learning and the use of an approximation of the Q-function as a neural network. The results are interesting, though dependent on the form of the reward function. Besides that, the initial learning with a predetermined policy appears less promising than an initialization using a precomputing of the Q-function through the traditional value-iteration algorithms in a model-based learning approach [11]. Amoui *et al.* have also applied RL successfully to the management of quality attributes in a news web application to optimize the application throughput [15]. An interesting point of this work is the use of simulation to initialize the learning functions.

Bahati and Bauer [16] propose to use RL to manage threshold-based rules. A first controller applies these rules to a target application in order to enforce its quality attributes. A second controller monitors these rules, adapts its thresholds to changing conditions and disables irrelevant rules. This approach is interesting since it limits the state space to appropriate state-action pairs and allows the reuse of learned models

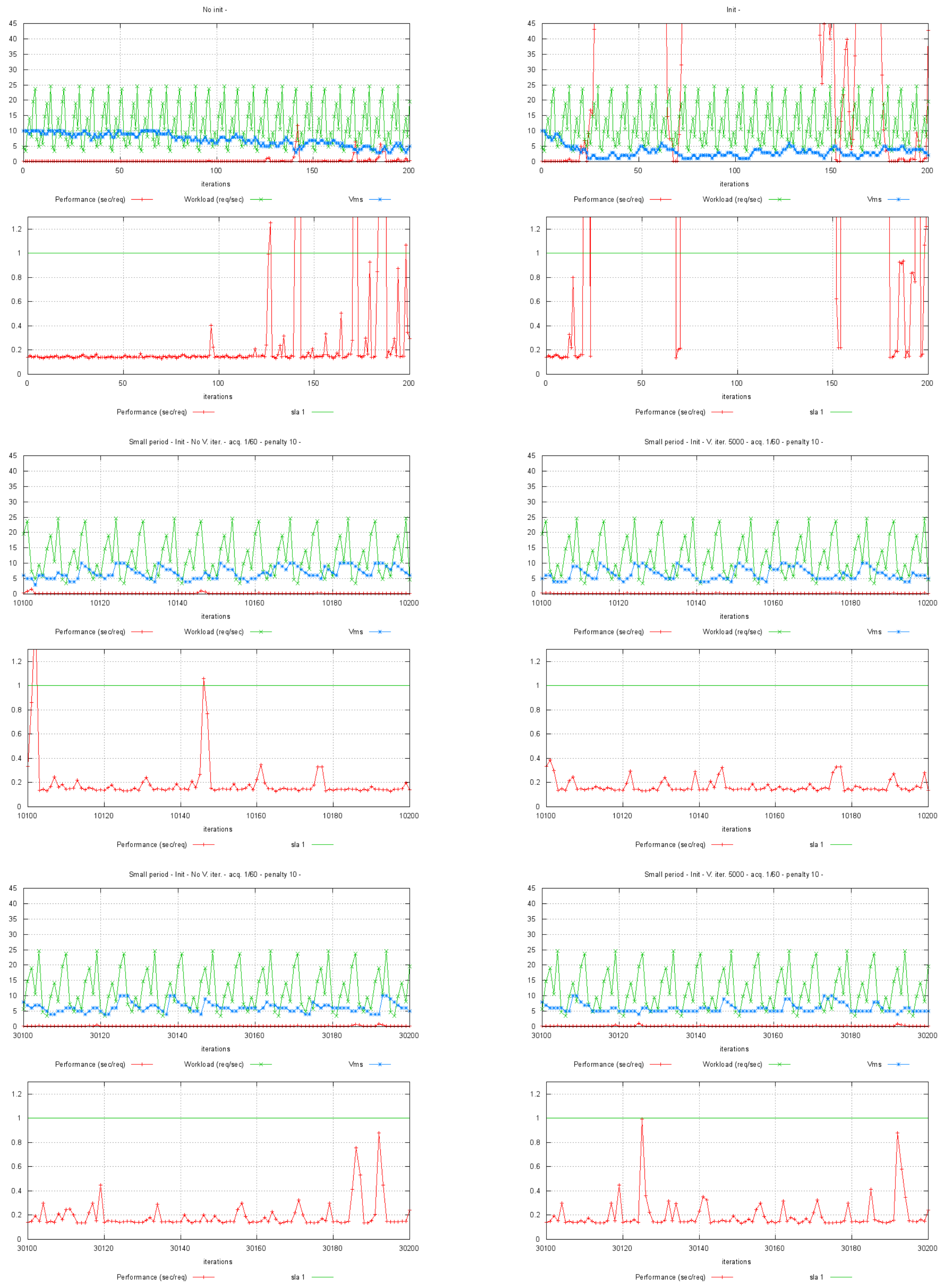


Fig. 3. Comparison between the decisions made during the learning process as observations pass given that convergence speedups are applied or not. On the left, plots show decisions and the resulting performance when no speedup is made, while on the right convergence speedups are applied every 5,000 observations.

from one rule set to the next. Nonetheless, this approach reintroduces the burden of setting up efficient thresholds, does not qualify learned models nor give much information about the time it takes to achieve convergence.

Zhang *et al.* propose a pragmatic approach to resource allocation which consists in preallocating enough resources to match up to 95% of the observed workload and then allocates more resources on another cloud when this threshold is passed [17]. No real automatic control approach is applied to prevent instability, however.

Kalyvianaki *et al.* design controllers built on top of statistical and Kalman filtering to track CPU utilization and allocate pieces of processing units to VMs [18]. Their agents scale individual VMs as well as multi-tier applications while coping with workload changes. However, authors limit themselves to vertical scaling within the bounds of three physical servers; their controllers only work on applications with immediate rewards and do not care about stability.

VI. Conclusion and Future Works

Reinforcement learning is a promising approach towards an autonomic solution to the problem of dynamically adapting the amount of resources allocated to applications in cloud environments. However, reinforcement learning algorithms require care and expertise to deal with the main requirements of self-adapting cloud infrastructures: good allocation policies from the start, prompt convergence to the optimal policy and capability to deal with evolution in the performance model of applications. In this paper, we have evaluated experimentally, through simulation, the possibilities of two techniques:

- Careful initialization of the learning functions in order to have a good policy from the start;
- Convergence speedups for model-based reinforcement learning which inserts complete policy evaluation steps at regular intervals into the learning phases.

We have simulated Olio, a standard testbed web application, on a cloud and applied to this simulation a reinforcement learning approach to resource allocation. The main conclusions from these experiments are:

- Good initialization made by evaluation standard policies proved successful to begin the learning and the decisions with an already good policy and therefore good performance from the start;
- Convergence speedup techniques did improve a bit the error in the Q-function of the learning process, but more importantly succeeded in making the learned policy approach the optimal one faster.

We are currently conducting experiments to validate the use of statistical testing over the behavior model learned to complete the more traditional use of continuous learning, to take care of large changes in the performance model of the application and the cloud infrastructure by reinitializing the learning process. We are implementing on a cloud prototyped at Orange Labs an automated workflow to put these techniques at work for the allocation of virtual machines to applications and therefore to provide with a truly autonomic solution to this problem on actual industrial-strength clouds.

Much work and experimentation still need to be done. Our experimentations to date show that, even with our proposed workflow, applying reinforcement learning on a per-application basis will still need more information from its environment to be usable in the real world. Reinforcement learning should rather be applied in the context of a larger-scale workflow, where clouds could gain information from applications to applications in order to make the techniques much more successful. We plan to experiment now with higher level descriptions of applications and their need for adaptation in order to select from past applications learned policies from which the learning can be initialized more accurately.

References

[1] X. Dutreilh, N. Rivierre, A. Moreau, J. Malenfant, and I. Truck, “From Data Center Resource Allocation to Control Theory and Back,” in *Proc. of the 3rd IEEE Int. Conf. on Cloud Computing, CLOUD 2010, application and industry track*. IEEE, 2010, pp. 410–417.

[2] R. Sutton and A. Barto, *Reinforcement learning — an introduction*. MIT Press, 1998.

[3] G. Tesauro, N. K. Jong, R. Das, and M. N. Bennani, “A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation,” in *Proc. of the 2006 IEEE Int. Conf. on Autonomic Computing (ICAC)*. IEEE Computer Society, 2006, pp. 65–73.

[4] J. Rao, X. Bu, C.-Z. Xu, L. Wang, and G. Yin, “Vconf: a reinforcement learning approach to virtual machines auto-configuration,” in *Proc. of the 6th Int. conf. on Autonomic computing (ICAC)*, 2009, pp. 137–146.

[5] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” Tech. Rep., July 2009. [Online]. Available: http://www.csrc.nist.gov/groups/SNS/cloud-computing/

[6] “Amazon EC2,” http://aws.amazon.com/ec2/.

[7] J. A. Rolia, L. Cherkasova, and C. McCarthy, “Configuring workload manager control parameters for resource pools,” in *NOMS*, 2006, pp. 127–137.

[8] D. Bertsekas, *Dynamic Programming and Optimal Control*, 2nd ed. Athena Scientific, 1995, volume 1 and 2.

[9] W. Sobel, S. Subramanyam, A. Sucharitakul, J. Nguyen, H. Wong, A. Klepchukov, S. Patil, O. Fox, and D. Patterson, “Cloudstone: Multi-platform, multi-language benchmark and measurement tools for web 2.0,” 2008.

[10] D. Bertsekas and J. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[11] M. L. Littman, “Algorithms for Sequential Decision Making,” Ph.D. dissertation, Dep. of Computer Science, Brown U., mars 1996.

[12] D. Sheskin, *Handbook of parametric and non-parametric statistical procedures*. Chapman and Hall, 2007.

[13] H. C. Lim, S. Babu, and J. S. Chase, “Automated control for elastic storage,” in *Proc. of the 7th Int. Conf. on Autonomic computing (ICAC)*. ACM, 2010, pp. 1–10.

[14] J. Xu, M. Zhao, J. Fortes, R. Carpenter, and M. Yousif, “On the Use of Fuzzy Modeling in Virtualized Data Center Management,” in *Proc. of the 4th Int. Conf. on Autonomic Computing (ICAC)*. IEEE Computer Society, 2007, pp. 25–34.

[15] M. Amoui, M. Salehie, S. Mirarab, and L. Tahvildari, “Adaptive Action Selection in Autonomic Software Using Reinforcement Learning,” in *Proc. of the 4th Int. Conf. on Autonomic and Autonomous Systems (ICAS)*. IEEE Computer Society, 2008, pp. 175–181.

[16] R. M. Bahati and M. A. Bauer, “Towards adaptive policy-based management,” in *NOMS*. IEEE, 2010, pp. 511–518.

[17] H. Zhang, G. Jiang, K. Yoshihira, H. Chen, and A. Saxena, “Resilient workload manager: taming bursty workload of scaling internet applications,” in *Proc. of the 6th Int. Conf. industry session on Autonomic computing and communications*. ACM, 2009, pp. 19–28.

[18] E. Kalyvianaki, T. Charalambous, and S. Hand, “Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters,” in *Proc. of the 6th Int. Conf. on Autonomic computing (ICAC)*. ACM, 2009, pp. 117–126.

Improving the Development Process for Teleo-Reactive Programming Through Advanced Composition

James Hawthorne, Richard J. Anthony, Miltos Petridis
School of Computing & Mathematical Sciences
The University of Greenwich
London, UK
{J.Hawthorne, R.J.Anthony, M.Petridis}@gre.ac.uk

***Abstract*—Teleo-Reactive programming as applied to autonomic systems is both a viable and exciting prospect as it inherently recovers from errors and unexpected events whilst proceeding towards its goal. It does this without the need to know in advance, the specific events which might occur, thus greatly reducing the human maintenance element. Whilst the benefits of this technique are great once the program has been composed, the challenge of validation attached to these programs also increases. We aim to ease the composition process needed when constructing T-R programs by building new abilities in to our existing Teleo-Reactive framework. Firstly, these new abilities will make it possible to detect validation issues at design time, thus reducing the likelihood of problems and increasing the ease of detecting them. Secondly, it will be possible for programs to be automatically composed from existing elements, thus further reducing the cost of validation issues. These ideas can even be extended to allow for dynamic composition and runtime changes to goals and actions.**

***Index Terms*—Teleo-Reactive Programming; Software Composition; Goal-Based Software**

I. INTRODUCTION

Teleo-Reactive (T-R) programs developed by Nilsson [1] were designed for autonomous control of mobile agents. T-R programs continually accept feedback from the environment, performing actions based on this current state. A T-R program is structured with a hierarchical list of condition and action pairs with each action fulfilling or partly fulfilling the condition of higher precedence. An action will end execution if it ceases to be associated with the highest true condition, either because the action has fulfilled the next condition or some other circumstance has caused this case. We have shown in previous work how T-R programs can be used to create reliable and robust autonomic solutions [2].

The Oxford English dictionary describes *Composition* as "the nature of something's ingredients or constituents; the way in which a whole or mixture is made up". This description could be imagined differently depending on the given allowable interaction of the context and the aim of the composition. For example, in a heavily component oriented software design [3], [4], most software engineers would imagine the links and interactions between the objects and components as being the *Composition* of the program. According to Szyperski [3], components "have to be carefully generalised to allow for reuse in a sufficient number of contexts" therefore the composition of the module could decide other labels it has attached. 'Reusable', 'modularised', or the antonym, 'procedural'.

When we talk about composition of T-R programs, the whole of the program is composed of its preceding rules leading to the satisfaction of the goal. The ordering of rules is fundamental in T-R programs as the higher the rule in the program, the higher the priority. In a procedural program, the modules of code are often tightly coupled to each other (if modules exist at all) and the composition of those modules are not greatly considered. The content of the modules take priority here. In component-heavy programs, the connections and interactions are considered to a greater extent, resulting in greater re-usability and portability. In this paper, when we talk about T-R composition, we are referring to the conditions each action is associated with and the order of the rules in the program list.

We have identified some of the problems in composing T-R programs [5]. It is easy to make a mistake in the logical design of the program such as incorrect rule ordering or an action needing an extra condition to proceed correctly. This could lead to perpetual skipping of one rule leading to falsehoods in another rule or that the program is deadlocked and can never reach its goal. In essence, T-R programs present a more natural and robust way of approaching a goal (natural in the sense that a human approaches an activity without first planning the millions of different events which could interrupt the process). This paradigm shift has however exposed new logic problems which are very easy to trigger and difficult to diagnose.

Very often in fact, a program may look entirely logical but when it is first run, mistakes become obvious to us. As an analogy, an incomplete diagram of a house may appear complete so an inexperienced builder may begin to build the house. However, a structural part of the house is not shown on the diagram and it is not obvious that it is needed either. After the house is built, the mistake is obvious.

In [5] we highlight some frequently occurring issues and propose a valuable set of guidelines to reduce the possibility of these issues from ever embedding themselves in the T-R program. We have already developed the Java Teleo-Reactive Autonomic Framework (JTRAF) to allow for the easy creation of high-level T-R solutions as well as allowing for future improvements and extensions such as the ones proposed in this paper to be applied without affecting existing JTRAF based programs. By building our solution into JTRAF, we can shift some of the validation work from the designer, tackle some of these logic problems and automate the T-R composition process. Thus T-R becomes a more viable autonomic solution.

Our proposal addresses two aims. The first, as already mentioned, is to detect validation issues with the program before deployment. This includes errors where an action causes two rules to become true at the same time. The lower priority rule is inadvertently made redundant because the highest priority rule is always executed before the lower one. Another error could be with deadlocked actions, where one action never causes a higher priority condition to change state and thus the logic can never proceed past this point. These are vitally important issues that are not easily detected or solved. From experience, we know that even simple T-R programs can contain problems which are not easy to diagnose through observation alone. Very often it is not until after an initial execution of the program, do we find that an action is deadlocked. This is more often than not, a fault in the high level logic than a code fault.

The second aim is a natural progression from the first. Namely, that if we can determine where faults are and if we can inform our program which actions contribute to the logical correctness of specific conditions, then it is intuitive that we should be able to automatically compose the program in terms of order and arrangement of rules and conditions. This will further reduce human maintenance costs.

II. Related Work

In [6] the authors describe a problem in T-R programs where actions are executed one at a time. They argue that actions should be eligible to be run concurrently with other actions as there may be periods where you do not want to cease one action to begin another. The work is proposed for the robotics domain for which T-R programs were designed. In one given example, "If a soccer robot should dribble and kick. If the robot stops dribbling in order to prepare kicking, the ball will roll away from the robot. Thus, the robot has to kick while it concurrently dribbles." If a robot has a similar T-R program:

$$\begin{array}{c} ..... \\ CanKick \longrightarrow Kick \\ HasBall \longrightarrow Dribble \\ ..... \end{array}$$

This may be a problem if the Kick action took more than a few microseconds to complete. In many scenarios it is possible to organise the logic so that actions are executed in a time-slicing / multi-process way, where two conditions are alternately switched between true and false states. The two actions would then give the illusion of concurrency.

Much of the work on T-R is aimed towards using a variety of artificial intelligence techniques to implement learning algorithms on the model. For example, [7] uses neural networks to capture new environmental experiences which can be integrated into a learning architecture. Whereas in [8] a representation formalism is developed called 'teleoreactive logic problems' which support learning. In this method, two knowledge bases exist containing a list of 'percepts' about the environment and a knowledge base containing known actions. They also describe an interpreter that utilizes the logic problems to achieve goals.

Through human guidance, the robots in [9] learn new T-R programs. These new programs are aimed mainly at navigation oriented tasks. To this end the low-level sensor readings are first transformed into higher level output so that they can be interpreted with physical objects represented as landmarks. Another learning algorithm is presented in [10]. The authors use genetic programming techniques to evolve T-R programs and produce programs for solving some problems.

We can think of a T-R system as a goal-oriented system as every T-R program proceeds towards its top level condition (the goal). There are several goal-oriented software designs which exist. Of particular note is [11], where a goal reasoning tool is built on the Tropos [12] methodology. Tropos itself supports software development at all stages, from requirements analysis to design with goal-based reasoning as a driving force. In [11], the authors try to make the goal analysis more complete by building forward and backward reasoning techniques into a graphical design tool.

This work is of particular interest to us because the backward reasoning employed is similar to the way in which our T-R software composition methods are implemented. The forward reasoning is employed to evaluate the impact of adopting the approaches with respect to *softgoals*. *Softgoals* in Tropos and indeed, in many other goal-oriented systems mean non-functional goals such as *Are customers happy?*.

In [13], the author argues the case for widespread use of Goal Oriented Requirements Engineering (GORE). He argues that it makes sense that the creation of software should be directed towards what the user wants to get from it, i.e. the goal, and that these goals should drive the requirements. The author shows evidence of several successful projects which use GORE and also argues that tool support should be integrated into GORE development.

[14] also champion GORE as a way forward in software development. They give a detailed account of the relationship between goals in GORE models and use several examples to illustrate their point. For example, the relationship between one goal and another goal or softgoal can have several satisfactory levels and contribution types, including satisfied, none, conflict, denied, some positive, some negative to name just a few.

It will certainly be interesting to see if our proposed techniques described in this paper can have their desired benefit with such minimal extra requirements on the T-R software designer considering the number of embellishments needed for the completion of a GORE-based system. However, the stages of development and their aims vary. i.e. GORE is largely

$$\begin{aligned} IsFileComplete &\longrightarrow Nil \\ IsFivePacketsSent &\longrightarrow ChangePacketSize \\ IsConnected &\longrightarrow SendNextPacket \\ IsAccepting \wedge IsWaiting &\longrightarrow Connect \\ T &\longrightarrow Accept \end{aligned}$$

Fig. 1. Typical view of a T-R program (simple file sender program)

concerned with modelling a system at the requirements stage whereas we want to aid the designer at the design/development stage.

Although T-R was developed as an autonomous low-level robotic and agent control method, we believe that the approach can be of use in higher level autonomic software and we are interested in applying this approach in this field. As such, we note some of this research.

There have been several approaches to autonomic software solutions, from architectural designs [15], [16] to artificial intelligence [17] and utility function based ideas [18], many of which are very complex solutions themselves and thus the ease of implementation is sometimes questionable. We wish to minimise "complexity tail chasing"; a term used in [19] to describe the situation where one aspect of the complexity problem is resolved by increasing complexity in other aspects. Such approaches can exacerbate the challenges of validation and trustworthiness.

III. Existing T-R Composition Method

For our discussion, we will use a very simple file sending T-R program first presented in [2]. This program was designed to show how the T-R system can recover from unexpected events. The program simulates a network file sending service from the point of view of the server. It sends a file in packets once a client has connected. Depending on the success of previous packet sending attempts, the packet size is adjusted accordingly; dynamically achieving a balance between communication efficiency and robustness.

A typical representation of this T-R program is shown in figure 1 with the boolean conditions to the left of the arrows, and the actions to the right. It is worth emphasising that the higher the rule, the higher the precedence. So lower rules are ignored if a higher condition is true. For example, in figure 1 if both *IsConnected* and a *IsFivePacketsSent* were both true then only action *ChangePacketSize* would be eligible to execute as *IsFivePacketsSent* is located higher in the program and therefore takes precedence (*IsFivePacketsSent* actually means has a *multiple of five* packets been sent 'true' or 'false').

This program was implemented using the JTRAF framework where each action contains only one condition. For example, the *Connect* action of figure 1 is instantiated in JTRAF with a reference to an instance of an *AndConditon* which itself contains references to instances of *IsAccepting* and *IsWaiting* conditions. Therefore connectAction only references one condition (*andCondition*).

```
                .....
Conditional andCondition = new AndCondition(
     new IsAccepting(), new IsWaiting());
```

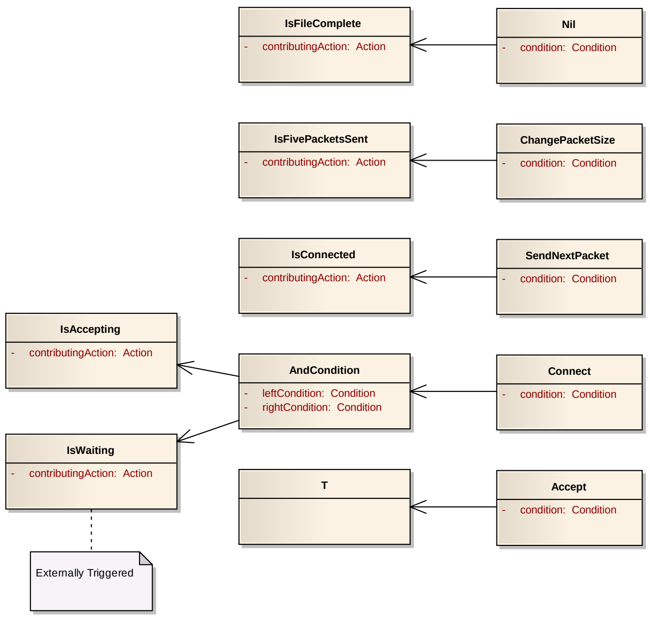


Fig. 2. Class diagram representation of the file sending program in JTRAF

```
Action connectAction = new ConnectAction(
     andCondition);
                .....
```

Each action is instantiated in a similar manner but does not yet include the sequence order and the T-R program still needs to be composed. For the simple file sending application the code is as follows.

```
                .....
TRProgram trProg = new TRProgram();
                .....
trProg.addActionToBottom(nil);
trProg.addActionToBottom(changePacketSize);
trProg.addActionToBottom(sendPacket);
trProg.addActionToBottom(connectAction);
trProg.addActionToBottom(acceptAction);
                .....
```

The method *addActionToBottom(Action)* is used by the program designer as shown to compose the actions in (what they perceive as) the correct order. In this case we create the T-R program of figure 1. A class diagram representation is shown in figure 2.

The important fact illustrated here is that, although each action has a reference to its conditions, it has no reference to any other part of the program. This could be considered good software design as the loose coupling means changes to one part of the program will have little effect on any other part.

A. Related Problems

The problem with T-R programs in general is the heavy reliance on the designer to correctly compose them. The order of rules and number of conditions related to each action and their arrangement have to be decided by the designer. This may appear straightforward but even in the simple file sending example mistakes can and were made. These mistakes not only invalidate the entire program but can be very difficult to find and diagnose.

As shown in figure 2 this heavy reliance is due to the fact that there is no link between the T-R elements within the program itself so the designer must arrange the elements in the order that is thought to be correct. What the designer thinks is correct and what is actually correct can be very different. Often, it is not until the program is first executed do these composition mistakes become evident.

As a real example, the program author (one of the authors of this paper) made the mistake of thinking that the packet size could be configured before packets were sent and originally reversed the order of the second and third rules. He neglected to recognise that organising the logic this way would make the packet optimisation rule redundant. As the packet sending rule now had priority over the packet optimising rule and since *sendPacket* only requires *IsConnected* to be true, the packet optimising rule will be skipped entirely. The goal of sending the file could still be achieved but without packet optimisation. Therefore it was difficult to ascertain that there was any fault in the program.

This example also serves to illustrate the difference in type of error which can occur in a T-R program and a program written in a high level language. We can imagine a similar program written in Java where any errors which do occur are likely to be a mixture of compile-time and run-time faults in code. Compile-time errors will be flagged by the compiler and corrected when they are first known. Run-time errors can be much harder to identify and are too numerous to have complete confidence that we have identified them all.

It is possible for an action to have an effect on more than one condition or a condition may or may not become true after only one iteration of its contributing action. If several iterations are needed; each successful iteration would be moving the program towards a state where the condition can become true. If a condition is made false by some action and is then prevented from ever becoming true again then the program is effectively deadlocked when *dropping* below this false condition as the associated action can now never execute and it may not be possible to reach the goal. Similarly, if a condition, once true, can never return to a false state then that may prevent a lower precedence condition from ever being reached. The associated action for the unreachable condition may be needed at a later stage by a different condition to make it true.

A T-R program can be correct despite not being able to predict its exact state at any moment due to its dynamic behavior which is sensitive to its operating context. This affords the T-R program the ability to recover from a broad range of unexpected events but the difficulty in building a formal model is exacerbated by the dynamic context-driven behaviour, as the sequence of states visited is not knowable pre-runtime and can be different in each run.

This highlights some of the complexity problems associated with composing a T-R program. Adding only one extra rule will likely result in an exponential growth in the complexity. The need for an automated method of detecting logical errors and composing T-R programs in order to expose T-R programming as a way to produce autonomic programs is proving to be a requirement rather than an option.

IV. Proposed T-R Composition Method

The T-R program designer will be required to make only one small addition to their program design. That is, to add a reference from conditions to actions informing JTRAF which actions contribute to the completeness of which conditions. By using this information to drive the composition algorithms we have implemented in the framework, JTRAF will be able to detect any logical errors in the T-R program design. With the newly supplied information and the current program composition it will be possible to determine if any condition is obtainable by recursively checking if the contributing action is located in a preceding rule. If so, we need to determine if this action will ever execute. We can do this by checking if the condition for this action is obtainable in the same manner as above. Essentially, we are forming a chain within the program as in figure 3 and thus be able to determine if the goal can be achieved or if any rule is skipped.

At the programming level we include a method (*isAchievable(Condition)*) which returns a boolean result indicating whether the provided condition will ever return true given the current composition of the T-R program. Since the goal of a T-R program is also a type of condition, the designer can test the goal and determine if the program is valid or not. If not, the designer can use the same method on the preceding conditions to determine the point of failure. For example in our previous file sender example, '*trProg.isAchievable(IsFileComplete);*' will return true because 'IsFileComplete' is achievable given the current program and additional contributing action information. However if the method returned false, the designer could work back through the program using this method in order to find the point of failure.

The *isAchievable(Condition)* method will warn the designer if for example the given condition 'is achievable' but some rules will be skipped entirely 'en route'. This indicates that there is an error in design but the condition is still able to become true.

As shown in figure 3 some conditions can be externally triggered. This means that the condition could be made true by some event external to the program itself. In the file sending case, *IsWaiting* becomes true when a client connects and is added to the waiting queue. Since we have no control within the T-R program over when this event might happen, we should mark the condition as being externally triggered. A condition should be marked as externally triggered if it is possible that the condition will become true at some point. A condition marked externally triggered when in fact it is always false could cause the program to be falsely validated as the goal might be perceived as reachable when in fact the incorrectly marked condition prevents this. In such a situation there is no guarantee that a condition will ever be triggered externally but this will not mean the program will ever fail

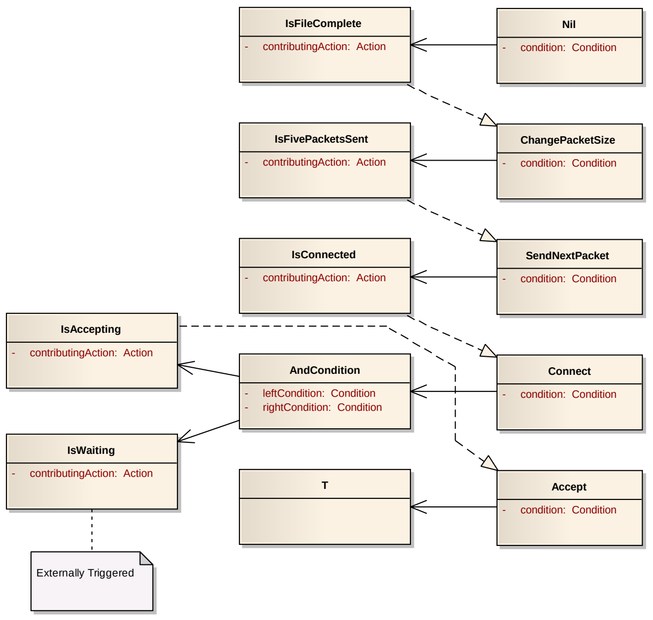


Fig. 3. Class type diagram representation of the file sending program in JTRAF with implemented composition design (solid lines indicate required conditions for an action, dashed lines indicate contributing actions for a condition)

entirely, only that the program will perpetually wait for the condition to become true.

The information required to detect composition mistakes is enough for the composition method to be extended, allowing JTRAF to automatically compose T-R programs as long as a composition is possible of course. Figure 3 shows the references between conditions and actions created through the contributing action variable and indicated by the additional arrows. The figure shows the program correctly composed and ordered. However, if the references exist, JTRAF will be able to reorder the rules and correctly compose the program even when the initial composition is incorrect or no previous composition exists. In any case JTRAF can return a correctly composed solution as in figure 3.

Again, at the programming level, a new method *compose()* will take the existing elements of the T-R program and arrange them so that the goal is achievable. Of course there may be more than one way to arrange the elements to satisfy the goal and the method may generate several failed compositions before a correct one is reached. The composition algorithm will implement the *isAchievable(Condition)* method as part of the mechanism for generating and testing for correct compositions. In this way the *isAchievable(Condition)* method becomes the foundation for composition, making *compose()* an intuitive and logical next step in validating a T-R program.

This short paper does not present the inner workings of the composition algorithm in detail, however, we provide a brief overview. For any condition to be achievable, it must hold a reference to a contributing action and the condition for this action must be satisfiable and associated with its own contributing action. This action must as well be associated with a satisfiable condition. This continues until the initial action and *T* condition are reached. If the solution is complete then it can be composed by JTRAF. This of course raises the possibility of multiple solutions, discussed in section VI.

Automatic composition has a big advantage for the program designer because it means that the logical ordering and numbers of conditions and actions and the problems of making a mistake here (see section III-A) are now less of a concern. If JTRAF cannot compose the supplied program, then it is likely to be because there is a lack of conditions or that the designer has neglected to add the contributing action to a condition. These problems are much simpler to rectify than the composition problems which occur without the aid of JTRAF's composition techniques. It also means that errors are discoverable at design time now. The designer does not have to wait until post deployment stage before composition problems are detected.

It is important to make clear that although information about which actions contribute to which conditions can now be provided, not every piece of information can be, and therefore the composition additions cannot guarantee the correctness of a program. Temporal data about condition checking and action execution might cause a deadlock but temporal data is not considered in JTRAF. This may cause a program to be validated whereas had the temporal data been considered and processed it would be impossible to do so. For example, suppose *ActionA* relies on *ConditionA* and *ConditionB* to be true. *ConditionA* is true when checked but it only remains true for two seconds. *ConditionB* takes three seconds before a true or false result is returned so by this time *ConditionA* is again false. If we knew this, then we could say that *ActionA* can never execute and therefore invalidates the program.

Entering temporal data into the composition algorithm would be of huge benefit in terms of validation for real-time applications but it would be unreasonable to expect precise timing measurements to be known in advance, especially for adaptive systems and/or systems with dynamic environments. A better solution would be for the measurements to be taken by and incorporated into JTRAF automatically. See section VI. It has always been an objective of JTRAF that any changes and improvements should not affect any existing programs which use JTRAF. Also, any additions which are warranted by the program designer should be easy to implement.

One of the main advantages of a pre-built Java framework is that the proposals in this paper and future methods can be seamlessly integrated into T-R designs. This means that the advantages are provided with the designer expending very little extra effort.

V. Conclusion

In this paper we have shown how effective and robust a T-R program can be and also how easy it is to compose the program incorrectly and thus never reach the goal. We have illustrated the seriousness of this problem and proposed that advanced composition techniques be built into our already existing Java T-R framework in order to automate the composition process, relieve the designer of this burden and to offer better assurance over the correctness of a program.

We believe that the techniques described in this paper are both necessary for the wide-spread use of T-R programming in both high and low-level applications. The solutions are implemented and benefits gained without requiring the program designer to endure much extra overheads. In fact to gain the benefits of advanced, automated composition, the designer is required to complete only one extra variable in order to link conditions to contributing actions.

VI. Future Work

There is a lot of scope here for future ideas to be implemented. In essence the work presented in this paper forms the foundation for many adaptations.

For example, if a condition has more than one action which can contribute to it; JTRAF could use this information to form multiple solutions for the completion of the program goal with the different solutions using the different possible actions. It may be that some of the actions contribute to the condition in question but not to another higher priority condition, which may be necessary for completion of the goal and therefore this particular solution is invalid. This information about the exact correctness of the composition might only be discoverable at run-time, since the program might need to be in operation before some information is known. For instance, the designer might claim that *ActionA* contributes to *ConditionA*, but after the program has been in operation, it is automatically discovered that this is not the case. JTRAF could then *self-adapt* according to an actual correct composition. The developers' original composition of the program could be treated as a suggestion rather than an unchangeable structure, with the composition always assessed and changed by JTRAF at run-time according to the proven reliability of each composition.

It may also be true that several of the actions are applicable to a particular program i.e. several actions could be used in the program to complete the same goal. In which case, JTRAF should determine which of the solutions is the better one. Perhaps one solution completes the goal in the quickest time whilst another solution is more reliable i.e. the actions in the reliable solution are much slower but rarely contain errors.

The failure of an action is less of a concern in a T-R program since if an action fails, all this means is that the program will fall back to a lower precedence rule until it is ready to try the action again. Never-the-less the program designer may require some actions to be more reliable than others. Perhaps using a combination of weights in a utility function and policies for the designer to decide whether speed or robustness is more of a concern to them.

We could also adapt JTRAF so that actions and conditions can be dynamically inserted into a running T-R program. Either a human could insert new conditions and actions or JTRAF could learn a new action by itself and dynamically insert it into the program. These newly learnt actions could even be used in the composition of completely new goals which were not possible before the new action existed. With the advanced composition techniques implemented there would be no need to decide where to place the new condition or action. JTRAF could decide where in the program it should be placed.

References

[1] N. J. Nilsson, "Teleo-reactive programs for agent control," *Journal of Artificial Intelligence Research*, vol. 1, pp. 139–158, 1994.

[2] J. Hawthorne and R. Anthony, "Using a teleo-reactive programming style to develop self-healing applications," in *Autonomics 2009: Third International ICST Conference on Autonomic Computing and Communication Systems*. Limassol, Cyprus: ICST, September 2009.

[3] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[4] C. Szyperski, "Component technology: what, where, and how?" in *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2003, pp. 684–693.

[5] J. Hawthorne and R. Anthony, "A methodology for the use of the teleo-reactive programming technique in autonomic computing," in *Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD), 2010 11th ACIS International Conference on*, June 2010, pp. 245 –250.

[6] G. Gubisch, G. Steinbauer, M. Weiglhofer, and F. Wotawa, "A teleo-reactive architecture for fast, reactive and robust control of mobile robots," in *New Frontiers in Applied Artificial Intelligence*, ser. Lecture Notes in Computer Science, N. Nguyen, L. Borzemski, A. Grzech, and M. Ali, Eds. Springer Berlin / Heidelberg, 2008, vol. 5027, pp. 541–550.

[7] J. Ramírez, "Neural synthesis of teleo-reactive programs," in *ICMAS '98: Proceedings of the 3rd International Conference on Multi Agent Systems*. Washington, DC, USA: IEEE Computer Society, 1998, p. 459.

[8] D. Choi and P. Langley, "Learning teleoreactive logic programs from problem solving," in *Proceedings of the Fifteenth International Conference on Inductive Logic Programming*. Springer, 2005, pp. 51–68.

[9] B. Vargas and E. Morales, "Learning navigation teleo-reactive programs using behavioural cloning," in *Mechatronics, 2009. ICM 2009. IEEE International Conference on*, 14-17 2009, pp. 1 –6.

[10] M. J. Kochenderfer, "Evolving hierarchical and recursive teleo-reactive programs through genetic programming," in *Programming, EuroGP 2003, LNCS 2610*. Springer-Verlag, 2003, pp. 83–92.

[11] P. Giorgini, J. Mylopoulos, and R. Sebastiani, "Goal-oriented requirements analysis and reasoning in the tropos methodology," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 2, pp. 159–171, March 2005.

[12] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, and J. Mylopoulos, "Tropos: An agent-oriented software development methodology," *Autonomous Agents and Multi-Agent Systems*, vol. 8, no. 3, pp. 203–236, 2004.

[13] A. van Lamsweerde, "Goal-oriented requirements enginering: A roundtrip from research to practice," *Requirements Engineering, IEEE International Conference on*, vol. 0, pp. 4–7, 2004.

[14] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the goal-oriented requirement language," *Int. J. Intell. Syst.*, vol. 25, pp. 841–877, August 2010.

[15] E. M. Dashofy, A. van der Hoek, and R. N. Taylor, "Towards architecture-based self-healing systems," in *WOSS '02: Proceedings of the first workshop on Self-healing systems*. New York, NY, USA: ACM, 2002, pp. 21–26.

[16] J. Kramer and J. Magee, "Self-managed systems: an architectural challenge," in *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 259–268.

[17] S. Hassan, D. Mcsherry, and D. Bustard, "Autonomic self healing and recovery informed by environment knowledge," *Artif. Intell. Rev.*, vol. 26, no. 1-2, pp. 89–101, 2006.

[18] J. O. Kephart and R. Das, "Achieving self-management via utility functions," *IEEE Internet Computing*, vol. 11, no. 1, pp. 40–48, 2007.

[19] R. J. Anthony, "Policy-centric integration and dynamic composition of autonomic computing techniques," in *ICAC '07: Proceedings of the Fourth International Conference on Autonomic Computing*. Washington, DC, USA: IEEE Computer Society, 2007, p. 2.

VCE - A Versatile Cloud Environment for Scientific Applications

Martin Koehler, Siegfried Benkner
Faculty of Computer Science, University of Vienna
Nordbergstr. 15/C/3, Vienna, Austria
koehler,sigi@par.univie.ac.at

Abstract—**Cloud computing promises to change the way scientists will tackle future research challenges by providing transparent access to distributed heterogeneous data sources and to high-end computing facilities for performing computationally demanding and data-intensive modeling, simulation and analysis tasks. In this article we describe the Vienna Cloud Environment (VCE), a service-oriented Cloud infrastructure based on standard virtualization and Web service technologies for the provisioning of scientific applications, data sources, and scientific workflows as virtual appliances that hide the details of the underlying software and hardware infrastructure. The VCE is based on a virtual appliance model within a component-based service provision framework, which supports the configuration of application, data, and workflow services from a set of basic service components providing capabilities like job or query execution, data transfers, QoS negotiation, data staging, and error recovery. The virtual appliance model constitutes an easy way to provision applications, data sources, and workflows in the Cloud and a standard way of accessing and integrating the appliances into client applications. VCE has been developed and utilized in the context of several projects for the realization of IT infrastructures within bio-medical and molecular modeling domains.**

Keywords-cloud infrastructure; software as a service; data mediation; workflow; MapReduce.

I. Introduction

Cloud computing is often defined as realization of the ”Everything as a Service” model(XaaS) [1]. Cloud computing promises on demand access to virtually infinite compute and storage resources, programming and execution platforms, and applications, provided and consumed as services. More often the cloud computing stack is characterized as Software/Platform/Infrastructure as a Service (SaaS, PaaS, and IaaS). IaaS Clouds provide a shared infrastructure to their customers on a rental basis by utilizing virtualization technologies while programming and execution environments for Cloud programming models (e.g., Google AppEngine, MapReduce [2]) can be categorized as PaaS. On the top layer, Cloud computing enables transparent and dynamic hosting of applications together with their native execution environment without knowledge about the actual hardware infrastructure. If applications themselves provide a direct service interface to the user, this approach is called SaaS. The emergence of virtualization technologies and Cloud computing infrastructures (e.g., OpenNebula, Amazon EC2) enables easy provisioning of software as virtual appliances on remote resources on demand. A virtual appliance can be defined as a software package preinstalled on a virtual machine image to enable provisioning of the software in the Cloud. For utilizing Clouds in the scientific domain, additional capabilities are required for supporting long running applications, usually executed on HPC computing resources, and for extracting knowledge from huge data sets.

In this work we present the Vienna Cloud Environment (VCE), which follows the Software as a Service model to expose virtual appliances as services. VCE virtual appliances hide the details of the underlying software and hardware infrastructure and provide a common set of generic interfaces to the user. The service provisioning framework in VCE has been developed on top of the Vienna Grid Environment (VGE), a service oriented infrastructure for virtualizing scientific applications and data sources as Web services. VGE [3][4] was developed in context of the European projects GEMSS and @neurIST and has now been Cloud-enabled by supporting virtual appliances.

VCE enables the hosting of parallel high-performance computing applications, data sources, workflows, as well as data-intensive MapReduce applications in the Cloud by means of specific virtual appliances, all following the same generic service interface. Virtual application appliances support on demand access to high performance computing applications (e.g., parallel MPI/OpenMP codes) including support for dynamic negotiation of service-level-agreements based on a flexible QoS infrastructure using business models specialized for the application [5]. In addition to virtual application appliances VCE provides virtual data appliances to facilitate access to and integration of heterogeneous data sources. Virtual data appliances are built upon OGSA-DAI and OGSA-DQP and offer transparent access to multiple data sources via a virtual global schema relying on flexible data mediation techniques [6]. On top of application and data appliances, VCE offers support for scientific workflows [7]. Workflow appliances are based upon the WEEP Workflow Engine [8] and can be structured to adaptively load balance the workload across multiple appliances. In recent work support for MapReduce appliances was implemented within VCE [9]. MapReduce appliances include an adaptive framework for the execution of data-intensive Map-Reduce applications in the Cloud based on autonomic computing concepts.

The remainder of this paper is structured as follows. The next section describes the architecture of VCE. The following sections present in detail virtual appliances for applications, data sources, workflows, and MapReduce applications. The paper concludes with a summary and an outlook to future work.

II. VCE Cloud Infrastructure

The VCE Cloud infrastructure comprises a set of virtual appliances, a generic service provision framework, and a client-side Cloud programming environment. The service provisioning environment enables service providers to expose compute intensive scientific applications, distributed data sources, scientific workflows, as well as MapReduce applications as Cloud services that can be securely accessed on-demand by clients over the Internet. Virtual appliances include a preconfigured setup of the VCE service provisioning environment as well as appliance specific software (e.g., Workflow Enactment Engine). The client-side Cloud programming framework offers a high-level application programming interface (API) with Java, C# and C bindings that may be used to construct advanced applications from application, data, workflow, and MapReduce services.

A. *VCE Architecture*

VCE adopts a service-oriented architecture and relies on standard Web Service as well as Cloud computing technologies for offering parallel applications, distributed heterogeneous data sources, workflows, and MapReduce applications as virtual appliances. VCE distinguishes four different types of virtual appliances, application appliances, data appliances, workflow appliances, MapReduce appliances, which all are based on virtual images, Web service enabled via WSDL and securely accessed using SOAP messages.

Application appliances virtualize compute intensive applications, usually parallel MPI codes available on clusters, other HPC systems, or Cloud resources. Using application appliances, clients may run applications on demand over the Internet and negotiate with service providers required QoS levels, for example, to ensure that an application job is completed before a certain deadline. VCE application appliances provide generic Web service components for job execution, monitoring, data staging, error recovery and application-level quality of service support.

Data appliances virtualize data sources as Cloud services, facilitating transparent access to and integration of heterogeneous data sources including relational data bases, XML data bases and flat files. Relying on advanced mediation mechanisms, data appliances provide transparent access to distributed data sources via a single integrated virtual schema. VCE data appliances provide generic Web service components for query execution, data movement, and staging of result data.

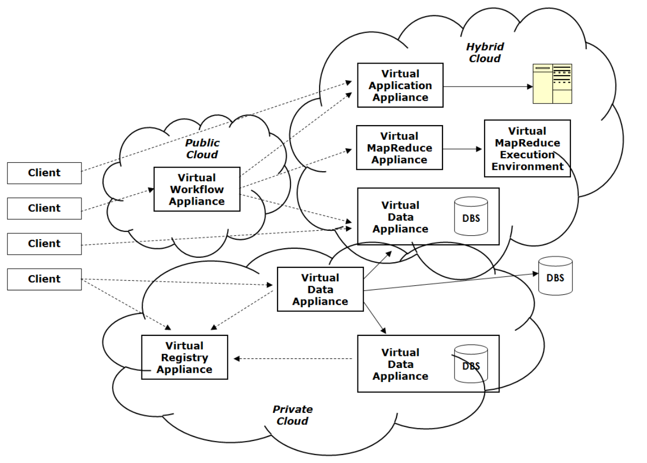

Hybrid Cloud
Virtual Application Appliance
Virtual MapReduce Appliance
Virtual MapReduce Execution Environment
Public Cloud
Virtual Workflow Appliance
Client
Client
Client
Client
Virtual Data Appliance
DBS
Virtual Data Appliance
DBS
Virtual Registry Appliance
Virtual Data Appliance
DBS
Private Cloud

Figure 1. A VCE Cloud comprising Application, Data, Workflow and MapReduce Services deployed in different Clouds

Workflow appliances virtualize scientific workflows as Cloud services, facilitating transparent access to and execution of scientific workflows. They are based upon the Workflow Enactment Engine (WEEP) and are able to schedule the workload to multiple workflow execution appliances. VCE workflow appliances provide generic Web service components for workflow execution, data movement, error recovery, and staging of result data.

MapReduce appliances virtualize data-intensive Hadoop applications, usually accessing huge data volumes stored in HDFS. MapReduce appliances internally use an adaptive execution framework based on autonomic computing concepts. The adaptive framework schedules the execution of the application automatically on available Cloud resources by considering the execution time as well as the number of resources. They provide generic Web service components for job execution, monitoring, data staging, and error recovery and are internally based upon the Hadoop framework.

As shown in Figure 1, a VCE Cloud usually comprises multiple application, data, workflow, and MapReduce appliances, as well as multiple client applications. Moreover, VCE offers an integrated certificate authority for providing an operational PKI infrastructure and end-to-end security based on X.509 certificates.

The VCE environment provides mechanisms for appliance discovery based on service registries. Multiple service registries may be set up in order to enable service providers to publish their services, and clients to discover these services. VCE service registries are realized as virtual appliances providing a Web service interface. Service providers can describe their services using an arbitrary set of attributes (name/value pairs), which are published in the registry. These attributes may be utilized during service selection to determine potential candidate services that might be able to fulfill a request.

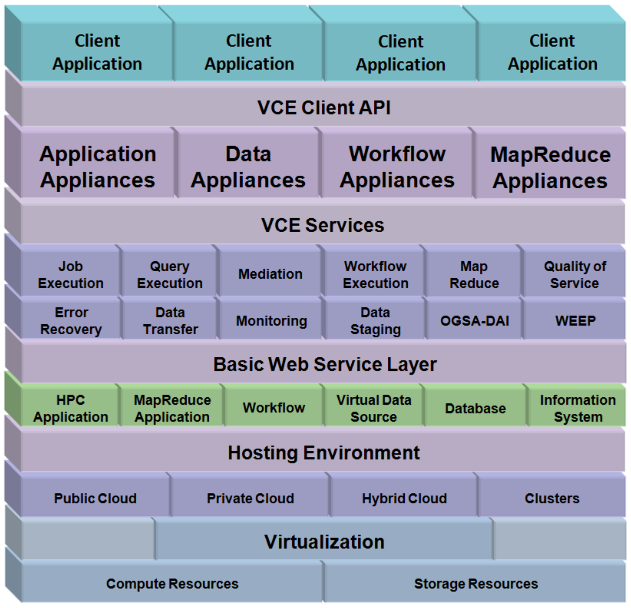


Figure 2. Layered abstraction of VCE

VCE virtual appliances, being virtual images, are hosted within Cloud environments. Different Cloud environments, as public Clouds (e.g., Amazon EC2) or private/hybrid Clouds can be utilized for the provisioning of virtual appliances. In several projects the following Cloud environments have been utilized: Eucalyptus, OpenNebula, and VMWare server. Virtual appliances provide access to one or more VCE services, which are hosted within a preconfigured service container. VCE client applications usually run on PCs or workstations connected to the Internet and make use of the VCE client API for interacting with services through the VCE middleware.

Figure 2 shows a layered abstraction of the VCE infrastructure. At the lower layer are a variety of compute and storage resources, which are Cloud-enabled via Cloud computing infrastructures or provided via schedulers as Sun Grid Engine (SGE). The HPC applications, scientific workflows, data sources and MapReduce applications are normally installed on VCE virtual appliances, but can be provided directly on the computing resources as well. These resources are virtualized through the Cloud service middleware layer and transparently provided through the abstraction of VCE services on virtual appliances, which are hosted on the Cloud resources. VCE services are composed of generic service components providing basic capabilities for job, query, and workflow execution, QoS, data mediation, data transfer, monitoring and error recovery. Client applications are able to transparently access VCE virtual appliances, through the abstraction of VCE services.

B. VCE Virtual Appliances

VCE virtual appliances provide a preconfigured installation of a service hosting environment based on the open-source frameworks Apache/Tomcat and Axis, a preconfigured VCE service, and a deployment tool for configuring and deploying VCE services. The VCE deployment tool automates the provision of HPC or MapReduce applications, data sources, and workflows as services. It offers an intuitive graphical or command line user interface for enabling service providers to describe, configure, deploy and manage services without having to deal with the details of Web service technologies.

The deployment tool enables to specify the service hosting environment, the security level, and a service description. The description of a service usually comprises the specification of input/output file names and of scripts for starting job execution, or the execution of a scientific workflow. A description of a data service comprises the specification and configuration of the underlying data sources. VCE services support different security levels, including no security, basic security via SSL, and end-to-end security. Supported security technologies include PKI, HTTPS, WS Security and an end-to-end security protocol for separate encryption of sensitive portions of the transferred data. The information specified by the user with the deployment tool is stored internally in an XML service descriptor. Upon deployment of a service, a Web service with a corresponding WSDL interface is generated, deployed in the VCE hosting environment, and published in the VCE registry.

A virtual appliance additionally includes an Apache server for connecting VCE services with the Internet using a Tomcat connector (JK connector) between the Apache server and the Tomcat server. Additionally the appliance includes a preconfigured firewall enabling access via ssh and http.

C. Virtual Appliance Access Model

VCE relies on a purely client-driven approach for accessing VCE virtual appliances via SOAP. All interactions of a client with appliances are initiated by the client and neither call-backs nor notification mechanisms are used.

VCE appliances are inherently multi-threaded, i.e., if multiple clients access a service, a separate thread is generated for each client, and for each client a separate application/workflow job or data access is generated. Session management and state handling is managed internally and transparently, conceptually following the WSRF model but being implemented based on conversational identifiers and WS-Addressing mechanisms.

A basic VCE appliance access scenario starts with administrative steps including authorization and authentication. The client then usually accesses a registry to find a set of candidate appliances. Afterwards the client uploads the input data, initiates execution, queries for the state of the execution, and finally downloads the result.

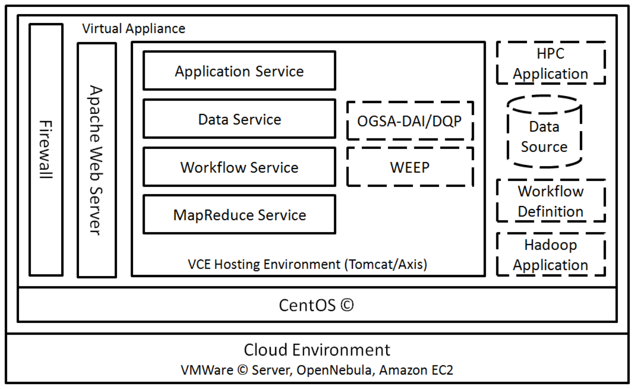


Figure 3. Architecture of Virtual Appliance

D. VCE Client Environment

In order to support the construction of client-side applications that interact with virtual appliances, VCE provides a high-level client API with bindings for JAVA, C and C# (.NET). The client API hides most of the details of dealing with remote services from the client application developer. The central abstraction provided by the client API is a ServiceProxy interface, which specifies all methods required for transparently accessing application and data services from a client application. Moreover, the client API provides a set of classes for dealing with various aspects of the VCE environment at a high-level of abstraction hiding the details of the underlying interaction with VCE.

The client API is structured into several layers. The bottom layers implement data marshaling, message generation, signing, and encryption, while the top layers provide abstractions for service discovery and all service interactions provided by the ServiceProxy. Using these high-level abstractions, users can more easily develop client applications that interact with VCE appliances. The ServiceProxy interface comprises methods for handling transfers of input and output data, for job, workflow or query execution, for monitoring, as well as for error handling.

Additionally, VCE clients support several security mechanisms as provided by VCE and monitoring of the execution. The high-level client API is based on the service component model and therefore extensible with new service component functionalities.

III. Virtual Application Appliances

The VCE enables service providers to virtualize HPC applications available on clusters, other parallel hardware, or virtual images as application appliances that can be accessed on-demand by clients over the Internet. Application appliances hide the details of their execution environment, providing abstract interfaces for managing job execution on remote computing resources. The VCE application appliance infrastructure has been partially developed within the EU Project GEMSS [10], which devised a service-oriented Grid infrastructure that supports the Grid provision of advanced medical simulation. In order to enable the utilization of Grid-based simulation services during medical procedures, QoS support to ensure the timeliness of simulation results was a major requirement. Addressing these issues, VCE application appliances may be configured with a flexible QoS negotiation service component, supporting dynamic negotiation of Service Level Agreements (SLAs) [5]. The QoS infrastructure enables clients to negotiate guarantees on service response times and price with service providers.

Virtual application appliances can be provided by directly installing the application on the appliance. In this case an application can be distributed to service providers by virtual application appliances and exposed as Web services by hosting the appliance in the Cloud. Virtual application appliances are able to utilize a scheduling system for the execution of jobs. Therefore it is possible to access applications installed on local cluster resources via Sun Grid Engine (SGE) as well as applications deployed on Cloud resources via VCE application appliances.

IV. Virtual Data Appliances

Virtual data appliances were partly developed in the context of the European project @neurIST [11], which developed an advanced service-oriented IT infrastructure [12][6] for the management of all processes linked to research, diagnosis and treatment development for complex and multifactorial diseases.

To support such scenarios, VCE includes a generic data management and integration framework for the provision and deployment of virtual data appliances. VCE enables the virtualization of heterogeneous scientific databases and information sources as Web services hosted in virtual appliances, which allows transparent access to and integration of relational databases, XML databases and flat files. The development of virtual data appliances utilizes advanced data mediation and distributed query processing techniques based on GDMS [13], OGSA-DAI [14], and OGSA-DQP [15].

The VCE offers virtual data appliances as well as virtual mediation appliances, all providing the same interface to clients. Virtual data appliances (VDA) provide access to a single data source, and virtual mediation appliances (VMA) offer transparent access to multiple data sources via a global virtual schema. The virtual schema of a VMA provides an integrated, global view of the underlying local data sources. Virtual mediation appliances translate queries with respect to the global schema into local queries, manage the access to the local data sources, and integrate results from local queries according to the global schema. Different, tailor-made views of distributed data sources may be provided for specific usage-scenarios, by setting up different variants of virtual mediation appliances.

Virtual mediation appliances can be composed recursively, i.e., a VMA can act as a local data source to another virtual mediation appliance, resulting in a tree-structured data integration scenario. In order to optimize complex data integration scenarios, virtual mediation appliances may be configured to support distributed query processing. For distributed query processing, the data mediation engine relies on OGSA-DQP, which has been integrated to coordinate the distributed execution of queries to local data sources utilizing a set of additional virtual evaluation nodes, as specified by the service provider.

Virtual data and mediation appliances are based on a virtual images and encapsulate a VCE service provisioning installation providing access to the underlying data source. A fully configured system installation is provided as a virtual machine, including preconfigured system components such as Apache, Tomcat and SSL configuration. This allows simple deployment for test and production use in Cloud computing infrastructures. Virtual data appliances includes a preconfigured data access service providing a Web service interface to a preinstalled MySQL database. Virtual mediation appliances include a preconfigured installation of a data mediation service with OGSA-DQP support. The service provider has to provide the mediation schema to the virtual appliance to configure the service. Both virtual appliances are available for VMWare and Eucalyptus and are based on CentOS.

V. Virtual Workflow Appliances

Support for virtual workflow appliances has been implemented in the context of the CPAMMS project. A major challenge in this project is the provisioning of seamless integrated support for scientific workflows in the domain of computational molecular modeling, which access HPC applications and are deployed on globally distributed computing resources. These scientific workflows are typically long running, deal with huge files, and have a need for dynamic execution control mechanisms. We offer support for scientific workflows in the VCE by seamlessly integrating the Ubuntu Cloud infrastructure supporting the scheduling of dynamic and partitioned workflows deployed on virtual workflow appliances. In [16], [7] we described virtual workflow appliances in detail and a case study workflow for computing photodynamics of biologically relevant molecules.

Due to the dynamic characteristics and because of the unpredictable runtime and resource requirements of this workflow, we implemented a framework for the provisioning of scientific workflows as virtual workflow appliances in the Cloud, where resources can be made available on demand. We integrated a workflow enactment engine into the VCE and prepared the middleware for hosting workflow services transparently in the Cloud. Following the Cloud and Web service paradigms, a virtual workflow appliance for hosting scientific workflows exposed as Web service has been developed. The virtual workflow appliance includes all required software packages allowing an easy deployment of new scientific workflows as Web services by leveraging Cloud technologies. Scientific workflows usually include many potentially long running scientific codes, each of them exposed as a VCE service, which may be invoked many times during a specific workflow. To make use of the dynamic resource allocation possibilities in the Cloud, our virtual workflow appliances are able to delegate these requests to basic service invocation appliances. This leads to a decentralized execution of the workflow in the cloud environment and allows to schedule the different service invocations to different instances of the appliance.

The basic service invocation workflow encapsulates the life cycle of accessing a VCE appliance and is deployed as WEEP workflow [8] in the basic service invocation appliance. A simple interface is provided by the service invocation workflow, which can be used by workflow designers during the workflow development process. The basic workflow constitutes an additional abstraction layer hiding the details of accessing and querying a specific VCE appliance from the workflow designer. Multiple instances of the basic service invocation appliance are hosted in the Cloud environment and can be used to realize a distributed execution of different service invocations.

The virtual workflow appliance can be used to deploy scientific workflows as VCE services without the need of additional software installation. The virtual workflow appliance is configured with a VCE service provisioning environment. The service provisioning environment includes a preconfigured VCE workflow service definition, which allows the deployment of a workflow service simply by the provisioning of a WEEP Grid workflow archive (gwa) including the BPEL process definition. Using the VCE deployment tool, the VCE workflow service can be deployed automatically by providing the reference to the used gwa. The deployed VCE service comprises a workflow service component, virtualizing a workflow behind the generic VCE interface, and a streaming service enabling direct file transfers between services based on a push/pull mechanism. The workflow service automatically deploys the provided gwa in a local WEEP Engine installation accessible via a JMX interface. A WEEP template service is instantiated for the gwa, and a engine service is created, when the workflow is started. It is possible to host several VCE workflow services in one virtual workflow appliance. The VCE workflow appliance is configured with an advanced WebPortal allowing live logging, online user management, and a push/pull mechanism supporting large data transfers directly between services.

VI. Virtual MapReduce Appliances

Virtual MapReduce appliances [9] enhance the VCE with support for data-intensive applications based on the Apache

Hadoop framework. Virtual MapReduce appliances include an installation of the VCE service provisioning environment and expose a VCE MapReduce service via a Web service interface. VCE MapReduce services use an integrated self-configuring and adaptive framework for optimizing the configuration of data-intensive applications at three abstraction layers: the application layer, the MapReduce layer, and the resource layer. For each layer generic descriptors, describing the evolving parameter set, are introduced. At the application layer the framework is able to configure the execution of parameter studies, which results in different resource needs of the application. The Hadoop framework provides a huge set of configuration parameters and the installation has to be optimized for the actual set of resources [17]. Using a MapReduce descriptor, the framework is able to automatically determine an optimized setting of Hadoop parameters, as for example the number of map and reduce tasks, on a per job basis. Leveraging a scheduler, in our implementation the Sun Grid Engine (SGE), allows describing and selecting the available computing resources by taking into account the actual resource needs for a job.

The adaptive framework evaluates the arising parameter set and self-configures all layers on a per job basis. It is based on the MAPE-K loop [18][19], which is a well-known concept from autonomic computing, and a utility function [20]. In recent work [9], we evaluate the possible configurations based on a history based performance model and a utility function. By adapting the utility function, the system allows to specify the main optimization goal, optimizing resource usage or optimizing the runtime of a job.

Virtual MapReduce appliances rely internally on a scheduling system for the execution of MapReduce applications on remote resources. In [9], we leverage the OGE to schedule applications on local cluster systems, but we additionally provide MapReduce appliances to enable dynamic hosting in Cloud environments.

VII. Related Work

Over the last years, the scientific community has undertaken a lot of effort for enabling the utilization of Clouds for science and for the provisioning of scientific Clouds. Some institutions provide access to their scientific Clouds at `ScienceClouds.org` [21]. They provide compute and storage capability to all scientists wanting to run their applications in the Cloud. On the other hand, there is a lot of effort in the creation of open source Cloud computing toolkits. These Cloud computing toolkits can be utilized by institutions for setting up private, public, or hybrid Clouds which can be utilized by domain scientists. The OpenNebula project [22] developed a software stack for the provisioning of Infrastructure as a Service (IaaS) Clouds. The VCE, can be utilized on top of IaaS Clouds, and provides an easy and generic way for Cloud-enabling scientific applications, data sources, and workflows as Cloud services following the SaaS concept. In [23], a discussion of benefits and issues regarding to science Clouds is provided. Identified as main benefits are the virtual ownership of resources as well as the ease of deployment, while some open issues include performance management, interoperability, and execution models.

The industry provides different Cloud computing solutions, most often on a pay per use basis. The Amazon EC2 Cloud [24] provides IaaS services to their customers based on different instance types. They support specialized services for HPC computing including Amazon Elastic MapReduce and Public Data Sets. Microsoft provides with Windows Azure [25] services for hosting and scaling of Web applications on their data centers. Google follows the PaaS approach by providing a high-level and scalable programming API to their users (Google App Engine [26]). Many more commercial service providers are supporting different aspects of the SaaS/PaaS/IaaS stack.

VIII. Conclusion and Future Work

In this paper we presented the Vienna Cloud Environment, a generic Cloud infrastructure for virtualizing HPC applications, data sources, scientific workflows, and MapReduce applications as virtual appliances in the Cloud. Virtual appliances hide the details of the underlying software and hardware infrastructure and can be easily distributed and deployed. VCE application appliances virtualize HPC applications running on clusters by providing common operations for transparently managing the execution of application jobs. Virtual data and mediation appliances address the complex problems associated with access to and integration of distributed heterogeneous data sources. Virtual workflow appliances add support for distributed execution of complex long running scientific workflows. They include a preinstalled workflow engine and allow the deployment and execution of composed workflows written in BPEL. A basic adaptive execution framework for MapReduce applications has been realized for virtual MapReduce appliances. By utilizing virtual appliances, scientists are enabled to run their applications on virtually infinite resources in the Cloud and to easily access them via a Web service interface.

Our future work will focus on improved adaptive execution mechanisms based on autonomic computing concepts addressing the trade-offs between execution time, resource requirements/costs for Cloud-based scientific applications.

References

[1] "Everything as a Service," March 2011, http://www.hp.com/hpinfo/execteam/articles/robison/08eaas.html.

[2] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[3] S. Benkner, I. Brandic, G. Engelbrecht, and R. Schmidt, "VGE - A Service-Oriented Grid Environment for On-Demand Supercomputing," in *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, November 2004.

[4] S. Benkner, G. Engelbrecht, M. Koehler, and A. Woehrer, "Virtualizing scientific applications and data sources as grid services," in *Cyberinfrastructure Technologies and Applications*, J. Cao, Ed. Nova Science Publishers, 2009.

[5] G. Engelbrecht and S. Benkner, "A service-oriented Grid environment with on-demand QoS support," in *ICWS-2009 - 7th IEEE Int. Conference on Web Services*, Los Angeles, CA, USA, 7 2009.

[6] M. Koehler and S. Benkner, "A service oriented approach for distributed data mediation on the grid," in *Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on*, Lanzhou, Gansu, China, August 2009, pp. 401–408. [Online]. Available: http://dx.doi.org/10.1109/GCC.2009.35

[7] M. Koehler, M. Ruckenbauer, I. Janciak, S. Benkner, H. Lischka, and W. N. Gansterer, "A grid services cloud for molecular modelling workflows," *International Journal of Web and Grid Services (IJWGS)*, vol. 6, no. 2, pp. 176 – 195, 2010.

[8] I. Janciak, C. Kloner, and P. Brezany, "Workflow Enactment Engine for WSRF-Compliant Services Orchestration," in *In The 9th IEEE/ACM International Conference on Grid Computing*, 2008.

[9] M. Koehler, Y. Kaniovskyi, and S. Benkner, "An adaptive framework for the execution of data-intensive MapReduce applications in the Cloud," in *DataCloud2011 - 1st Int. Workshop on Data Intensive Computing in the Clouds*, Anchorage, Alaska, USA, 5 2011.

[10] S. Benkner, G. Berti, G. Engelbrecht, J. Fingberg, G. Kohring, S. Middleton, , and R. Schmidt, "Gemss: Grid-infrastructure for medical service provision," in *Proceedings of HealthGRID 2004*, January 2004.

[11] H. Rajasekaran, P. Hasselmeyer, L. L. Iacono, J. Fingberg, P. Summers, S. Benkner, G. Engelbrecht, A. Arbona, A. Chiarini, C. Friedrich, M. Hofmann-Apitius, B. Moore, P. Bijlenga, J. Iavindrasana, H. Müller, R. Hose, R. Dunlop, A. Frangi, and K. Kumpf, "@neurIST - Towards a System Architecture for Advanced Disease Managment through Integration of Heterogeneous Data, Computing, and Complex Processing Services," in *IEEE International Symposium on Computer-Based Medical Systems*. Jyväskylä, Finland: IEEE Computer Society Press, June 2008, copyright (C) IEEE Computer Society.

[12] S. Benkner, A. Arbona, G. Berti, A. Chiarini, R. Dunlop, G. Engelbrecht, A. F. Frangi, C. M. Friedrich, S. Hanser, P. Hasselmeyer, R. D. Hose, J. Iavindrasana, M. Köhler, L. L. Iacono, G. Lonsdale, R. Meyer, B. Moore, H. Rajasekaran, P. E. Summers, A. Wöhrer, and S. Wood, "@neurist: Infrastructure for advanced disease management through integration of heterogeneous data, computing, and complex processing services," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 14, no. 6, pp. 1365–1377, November 2010.

[13] A. Wöhrer, P. Brezany, and A. M. Tjoa, "Novel mediator architectures for grid information systems," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 107 – 114, 2005.

[14] M. Antonioletti, M. Atkinson, R. Baxter, A. Borley, C. Hong, P. Neil, B. Collins, N. Hardman, A. C. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead, "The design and implementation of grid database services in ogsa-dai: Research articles," *Concurrency and Computation : Practice and Experience*, vol. 17, no. 2-4, pp. 357–376, 2005.

[15] M. N. Alpdemir, A. Mukherjee, A. Gounaris, N. W. Paton, P. Watson, A. A. Fernandes, and D. J. Fitzgerald, "Ogsa-dqp: A service for distributed querying on the grid," in *Advances in Database Technology - EDBT 2004*, ser. Lecture Notes in Computer Science, E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Böhm, and E. Ferrari, Eds. Springer Berlin / Heidelberg, 2004, vol. 2992, pp. 3923–3923, 10.1007/978-3-540-24741-8_58.

[16] M. Ruckenbauer, I. Brandic, S. Benkner, W. Gansterer, O. Gervasi, M. Barbatti, and H. Lischka, "Nonadiabatic Ab Initio Surface-Hopping Dynamics Calculation in a Grid Environment - First Experiences," in *Proceeedings of the 2007 International Conference on Computational Science and Its Applications (ICCSA 2007)*, ser. LNCS, vol. 4705. Kuala Lumpur, Malaysia: Springer Verlag, 2007, pp. 281–294.

[17] Impetus, "Whitepaper: Deriving intelligence from large data using hadoop and applying analytics," March 2011.

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

[19] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, pp. 7:1–7:28, August 2008. [Online]. Available: http://doi.acm.org/10.1145/1380584.1380585

[20] W. Walsh, G. Tesauro, J. Kephart, and R. Das, "Utility functions in autonomic systems," in *Autonomic Computing, 2004. Proceedings. International Conference on*, May 2004, pp. 70 – 77.

[21] "ScienceCloud," March 2011, http://scienceclouds.org.

[22] "OpenNebula," March 2011, http://opennebula.org.

[23] C. A. Lee, "A perspective on scientific cloud computing," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, ser. HPDC '10. New York, NY, USA: ACM, 2010, pp. 451–459. [Online]. Available: http://doi.acm.org/10.1145/1851476.1851542

[24] "Amazon elastic compute cloud (amazon ec2)," March 2011, http://aws.amazon.com/ec2.

[25] "Windows Azure - Microsoft's Cloud Services Platform," March 2011, http://www.microsoft.com/windowsazure/.

[26] "Google AppEngine," March 2011, http://code.google.com/appengine/.

Self-adaptive tuning of dynamic changing problem solving: a first step to endogenous control in Multi-Agents Based Problem Solvers.

Gaël Clair, Frédéric Armetta, and Salima Hassas
Laboratoire GAMA,
Université Lyon 1
Lyon, FRANCE
Email: (gclair ; farmetta ; hassas) @univ-lyon1.fr

***Abstract*—In this paper, we propose a new model, EC4MAS, for endogenous control in self-organizing system for problem solving. We first present the characteristics of exogenous and endogenous control and draw the differences between them. Problem solving methods are often faced with the exploitation/exploration dilemma. We propose in this work an approach that tries to find a good coupling between structural/topological characteristics of a problem and the local associated behaviors that compose the solving method. More precisely we organize the multi-agents based solver into a social organization that represents the organization of the different local behaviors and a spatial organization that represents the different characteristics of the problem. The objective of the system is thus to find a good coupling between these 2 organizations. We illustrate our approach on a graph coloring problem, show that our model can solve the exploitation/exploration dilemma in this case and how it can provide a mean to build a more robust and efficient tuning of the solving process on dynamic changing problems.**

***Keywords*-multi-agent system; self-organization; endogenous control**

I. Introduction

The design of self-organized multi-agent systems (MAS) raises the issues of developing a decentralized control and the fitness of the global (eventually emergent) behavior produced through multiple interactions of various local behaviors. As in these ascendent approaches, the understanding/description of the global behavior is not reducible to the understanding/description of its composing local part, it is also difficult to imagine an easy way to control the global system, by controlling its local components. Much work have been achieved to address this issue, either during design or run time [1][2][3]. They are often designer's knowledge or heuristics based, which makes them often problem specific and hardly adaptable to complex dynamic problems.

In addition, complex problem solving is also subject to the exploitation/exploration dilemma when searching the solution space. When and how to explore is primordial to obtain a good quality of the solving process. A badly tuned exploration process can lead the system into local optima, or can be the source of uncontrolled perturbations [4]. This difficulty is increased in dynamic problems where permanent changes of the solution space occur.

We propose in this paper a new model for endogenous control in self-organizing multi-agents based complex problem solvers. Our model deals with the dynamic of the solving process. More specifically, we propose a process that dynamically couples the problem structural/topological characteristics to the appropriate solving behavior. The global solving behavior is achieved by a society of agents on an environment that represents the problem to solve. Each agent of the society achieves a specific local behavior, on a specific situation of the environment representing a structural characteristic of the problem to solve. The MAS has thus to find a social organization, a non random layout of local behaviors, that best fits a spatial organization, a non random layout of local configurations, representing the structural/topological problem characteristics. The endogenous control is the mechanism that allows the dynamic coupling of these two organizations, as a solution for the problem solving. In this paper, we describe our model based on this dynamic coupling and we show how it deals with the exploitation/exploration dilemma and the associated dynamic parameters tuning for a graph coloring problem.

Section II is a related work section about control in multi-agents systems, distinguishing exogenous and endogenous control and drawing the differences between them. Section III describes the exploitation/exploration dilemma. The proposed model is then defined in Section IV. Section V presents some experiments and their discussed results on a graph coloring problem. Section VI concludes the presented work and draws some perspectives.

II. Related works: Control in self-organized systems

In this section, we distinguish exogenous and endogenous control. In the exogenous case, the control is supported by mechanisms that are outside the system, whereas in the endogenous case, control mechanisms are issued by the inside of the system.

A. Exogenous Control

When the control system is based on external definitions, like fixed rules for example, it is qualified as exogenous control. Exogenous control is mainly dependent on the multi-agent system structural characteristics and is often based on the designer or user's knowledge. Using a classification given in [5], we describe in this section some approaches for control in MAS : Design based approaches and Calibration approaches. In Design based approaches, the control is expressed through the process design, by defining the global behavior at macro level and operational rules for its realization at micro level [1]. The relation between macro and micro levels is fully addressed at design time like in AALAADIN [2] or in ADELFE [3]. Calibration based approaches work on reducing the size of the search space, either by using knowledge about the problem to solve [6] or dividing the search space when the parameters are independents [7]. With design approaches the control is relatively static in the sense that once defined it is not easy to make it evolve during computation. In dynamic control approaches, the control is achieved through the guiding of the system's behavior at run time. The control system has a target to achieve/to maintain, means of action on the system and means of observation of the system. Two kind of dynamic control approaches are distinguished: *a priori* control and *a posteriori* control.

In a *priori control*, a set of possible agents's actions are predefined and used to determine the most adapted individual behavior, using tools like Markov Decision Process (MDP) [8]. The relations between global patterns and local behaviors are treated during the learning stage of the MDPs. A global *a posteriori* control approach is presented in [9]. The idea consists to define a system with a correct behavior and then influence this behavior through external control. This global state is captured as a pattern representing the aggregation of all the states of the different agents. If the form is not adequate, it is corrected by acting on the agents local behaviors. The forms definition and their evaluation are determined by preliminary simulations and user's knowledge. Assuming that the observed shape can be reduced to the aggregation of agents' states is too restrictive because it ignores interactions and the system's dynamics. This morphocontrol is an *a posteriori* control because it is applied during the system's computation. Control actions are not totally known at the beginning of system's computation.

B. Endogenous Control

To go further through an endogenous control, the controlling process should guide the agents behavior, but do so in a comprehensive manner. The control system must be built and evolved during the system computation to ensure that it is always consistent with the current situation. An endogenous control needs to be designed as an internal process to the system to control, and in an unsupervised manner. Three important concepts may provide building blocks for a controlling system: learning, observation and dynamics.

- Learning is an important step in a controlling process. The system needs to "understand" its past behavior to know which selected actions have produced success or failures. In endogenous control, the problem structure is learned during computation without any intrusion from outside the system. Learning is strongly related to the systems dynamics including the different configurations encountered. Unlike exogenous control, these configurations are not created artificially, but from real computations. Learning is more restricted but is continuous and provides a greater diversity in the situations addressed. To learn from their actions, agents must have some means to evaluate their environment and means to register the interests (good or bad) of their past actions to update the control system.
- Observation is important as far as it could be used to assess the current situation and then act accordingly. When one looks at the overall behavior of the system, one first seeks to determine the situation in which the system is but also indirectly determines the agents' states. The overall configuration, as we have already stated, is not only dependent of agents local behaviors but also of their interactions. Thus, the process that allows to understand the dynamics that lead to the overall configuration of observation should not be centralized and static. If the system must determine its state, it does not has access to a global observation, unless there is a centralization (or synchronization), which is not acceptable in decentralized autonomous system. The observation should be limited but should not be purely local. It should provide a broader vision about agents local behaviors and their organization.
- Lifelong learning implies a permanent dynamics in the system of control and influence. Modifying the control data influences also its actions. This development is a form of permanent adaptation to control the situation. This dynamics is an important element in the endogenous control. Beside this internal dynamic, the external dynamic has to be considered. The multi-agent system is situated in a permanently evolving environment, so the system is always submitted to perturbations and has to adapt itself. This dynamic does not directly direct the multi-agent system but it influences the control system. The perturbations are captured and interpreted by the control system, then it adapts itself to direct the agents' reactions.

The three previous elements are strongly related to each other. To learn from the computation it is necessary to be able to observe the evolution of the system and to understand its dynamic. In order to create an endogenous control, local observation and evaluation means of the global state, means to share self interpretations and means to update the control system are the elements to study.

III. Exploitation / Exploration dilemma

To illustrate our view of an endogenous control in a multi-agent system, we will use in the next sections, the exploitation/exploration dilemma. Exploration aims to gather as much information on the solution space. Exploitation intensifies the search around selected areas already encountered, based on information collected through exploration.

The exploitation/exploration problem can be divided into two sub-problems, which are *when to explore* and *how to explore*, which we discuss in this section.

The *when to explore* problem means that the system at a given time has to determine whether it is more useful to keep exploring the solution space to collect more details, or to improve already encountered solutions that seem promising. This problem can also be divided into two types, directed and undirected decision. In undirected methods the exploration is chosen randomly at any time of the search. In directed methods, the decision to explore is triggered based on previous knowledge of the problem. This type of methods use recorded past experiences to direct the search. Directed methods are typically used in reinforcement learning [10] to find an adapted exploring rate.

The *how to explore* problem determines the strategy to use to explore when exploration had been selected. The efficiency of a search is very dependent on the exploration process because it introduces some perturbations in the solving process. On the one hand these perturbations can be very useful when the search is stuck in local minima, on the other hand it can be very problematic when the process is close to a solution. The jump due to the exploration has to be adapted to the current search strategy and the solution space. [4] shows the influence of the exploration strategy comparing gentle and strong exploration strategies, which determines the size of the jump done in the solution space.

These two problems are very important to study when we want to build a complex problem solving system like a multi-agents based one. In the case of a self-organized multi-agent system, where control is decentralized and autonomous, it is not easy to memorize and use a big amount of previous information. The local action, evaluation, system dynamics and decentralization characteristics prevent the agent to easily learn from its own action. Exploit/explore strategies in autonomy can be found in [11]. An endogenous control can be a good tool to solve the exploitation/exploration problem, because it is created during the computation and is based on it. In order to illustrate our model for endogenous control, we will focus on the exploitation/exploration problem in a graph coloring problem.

IV. Endogenous control for multi-agent system

We propose in this work, a new model for endogenous control in self-organizing multi-agents based complex problem solvers. Our model deals with the dynamic of the solving process. More specifically, we propose a process that dynamically couples the problem structural/topological characteristics to the appropriate solving behavior. The global solving behavior is achieved by a society of agents on an environment that represents the problem to solve. Each agent of the society achieves a specific local behavior, on a specific situation of the environment representing a structural characteristic of the problem. The multi-agent system has thus to find a social organization, a non random layout of local behaviors, that best fits a spatial organization, a non random layout of local configurations, representing the structural/topological problem characteristics. The endogenous control is the mechanism that allows the dynamic coupling of these organizations, as a solution for the problem solving.

In this section we define the three key elements of the proposed model for endogenous control in multi-agents system (EC4MAS) which are the social organization, the spatial organization and the coupling. The control system has to represent the influence between the main system and its environment. The model is based on the relationship of three elements to construct an emergent control in the MAS to ensure its permanent and continuous adaptation.

A. Social organization

The multi-agent system involving multiple interacting agents, must be addressed in a more extended view than of a single agent. The overall activity is dependent on all individual actions but also on the interactions between agents. The group of agents is a reflection at a given time of the search activity of the system. This activity has to be captured by the system and has to be used to direct and control the research. To implement this perception we use :

- $RSo = \{Rso_1, ..., Rso_n\}$, a set of n social roles
- $Lso = \{Lso_{11}, ..., Lso_{nn}\}$ where $Lso_{ij} = (Rso_i, Rso_j), \forall i, j \in [1, ..., n]$
- $\forall Lso_i \in Lso, Cso = \{Lso_1, ..., Lso_i\}$, a set of social contexts
- a social organization $Oso =< Rso, Lso >$

The roles Rso act as guides for the agent to determine the appropriate strategy and dictate it a predefined behavior. The adoption of a social role by an agent implies that it adopts the guidelines and directives of that role. The action of the agent, so its social role choice, could have been influenced by the other agents social roles, this result in the relation Lso_{ij}. A relation Lso_{ij} exists if an agent with the role Rso_i has encountered an another agent with the role Rso_j. The social organization involves a set of roles Rso and relations Lso between them. The activity of a single agent can not be isolated from other agents and therefore the social roles are linked within the organization. The social organization Oso gives information of the agents situation within the solving process at a given time. The situation of an agent, and more precisely the adequacy of its social role, is directly dependent on its environment. To locate an agent of a social perspective, relations between him and its neighbors define a context Cso. The context is a part of the social organization, with a limited size around one particular agent, it provides information on relations between different social roles So in a particular situation of the resolution.

Formally, the social organization is used to represent the solving strategy of the system. Social roles' relations are based on the activity of the system, and more particularly on the agents' actions. This dynamic updates permanently the organization to adapt the search.

B. Spatial organization

The spatial organization is used to model the current situation in the search space, it is based on :

- a set of m spatial roles $Rsp = \{Rsp_1, ..., Rsp_m\}$
- a configuration $Csp = \{a_1, ..., a_j\}$ with $Csp \in SP$ where a_i is an agent state and SP is the search space
- a function $fRsp : SP \rightarrow Rsp$

The spatial role Rsp represents some characteristics of the current solution in the space of solution. It reflects the current position of the agent in the search space and allows it to apprehend the difficulty of its situation. The organization of a group of agents in the environment or physical agents organizations Csp, can highlight basic characteristics of the problem relevant to the resolution. In order to capitalize these informations, it is necessary to allow their identification and use by the system. The spatial organization is based on the $fRsp$ function which associates a spatial role to an agent from the current spatial configuration. This function uses sensors, given to the agents to capture their situation, to determine the Rsp. The sensors could be specific to the problem to solve or more generic as we will see in V-B2.

Unlike social information which models the actions of the agents, spatial roles are about the effect of the solving process in the physical environment. The role is essentially descriptive of the problem and the situation of the system during the resolution. The spatial organization connects particular configurations of the problem. In some cases the problem definition may give access to specified elements to define spatial roles such as topology of the graph for graph coloring. In other cases this information is not identifiable from the outset but may appear during the resolution.

C. Coupling

Social and spatial organizations both provide information of a different nature. The first one is particularly interested in the mechanisms of resolution looking to the fittest agents' behavior. The second one gives information on the status of the system in search space. These two elements are strongly linked because the social role defines how they act in the environment and the spatial role represents the situation of the system in the environment. The coupling of these two organizations is defined by :

- a coupling function $fC : Cso \times Rsp \rightarrow \mathbb{R}$ with $\forall x \in Cso$ and $\forall y \in Rsp$, $0 \leq fC(x, y) \leq 1$
- a fitness function $fT : (Cso \times Rsp) \times Time \rightarrow \mathbb{R}$ where $Time$ is the number of cycles of the resolution

The coupling fC is dynamic and allows the relations between space and social roles to evolve according to the fitness function fT evolution. The determination of the best couple *(Rso, Rsp)* for an agent in a given situation, is the key to success. This coupling is determined by evaluating and storing the pairs created by agents during the search in the previous cycles (a cycle is an amount of time where each agent acts one time). A look back at previous choices with fT allows to update the coupling fC to adapt the control system.

D. EC4MAS principle

Social roles are based on the actions we want the agents to do. They implement the mechanisms used to solve the problem, like explore or exploit strategies for example. Spatial roles are defined in order to give some specific informations on the search space. Spatial sensors have to be able to capture the current situation in the solution space. The fitness function allows the evaluation of the evolution of the search in time.

Algorithm 1 Agent cycle

```
$UpdateCoupling(fT);$
$Cso \leftarrow GetSocialContext();$
$Rsp \leftarrow GetSpatialRole();$
$Rso \leftarrow GetSocialRole(Cso, Rsp, fC);$
$Act();$
```

In an agent action cycle, given by algorithm 1, the agent uses the fitness function to update the coupling values then it gets its social and spatial situation to get its new social role according to the coupling function.

V. Results and discussion

In this section, we present an example of the use of EC4MAS to solve a graph coloring problem and we will focus on the exploit/explore problem in graph coloring.

A. Experimental Setup

The agents of the system represent the nodes of the graph to color. A solution is found when all the agents have no conflicts with their neighbors, so each two connected nodes are assigned different colors.

The main solving strategy of the agents is based on the Min Conflict heuristic. Two social roles are used, the first is the exploitation strategy and the second is the exploration strategy. Exploitation tends to decrease the number of conflicts between an agent and its neighbors. Exploration can randomly take a color or apply the exploitation strategy (Min Conflict with exploration). The social organization is modeled with a tree, where one role is represented at a level. The children are based on the representation amount percentage of the role in a situation. The figure V-A-1 illustrates a social organization where relations between roles is divided into two possibilities.

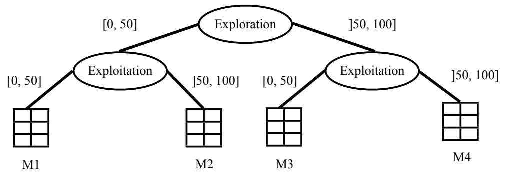


Figure 1. Example of a social organization

The spatial roles are based on the nodes degrees (static). Each role regroups nodes with similar degree. The spatial organization follows the graph topology.

The coupling is done using matrices to link social and spatial organization. The matrices M1, M2, M3, M4 in figure V-A-1 have one column per social role and one row per spatial role, so in this case there is 2 social roles and 3 spatial roles. The values in the matrix are float numbers between 0 and 1 and are normalized on the row. To use the coupling an agent finds its spatial role and its social context, so the associated matrix to use to choose a social role. Higher is the value in a social column, higher is the probability for the social role to be selected.

B. Results and analysis

To test our model we generated 100 different graphs with different seed. We used equi-partite 4-colorable graphs with 300 nodes, that means that the 4 color sets have the same size or can only be different from one node. An edge connectivity (*ec*) of 0,02333 (7/300) is used to get hard problems like seen in [12]. Each resolution is done 1000 times with a maximum of 1000 cycles. The performance is the number of cycles to get a solution, if no solution is found 1001 is used.

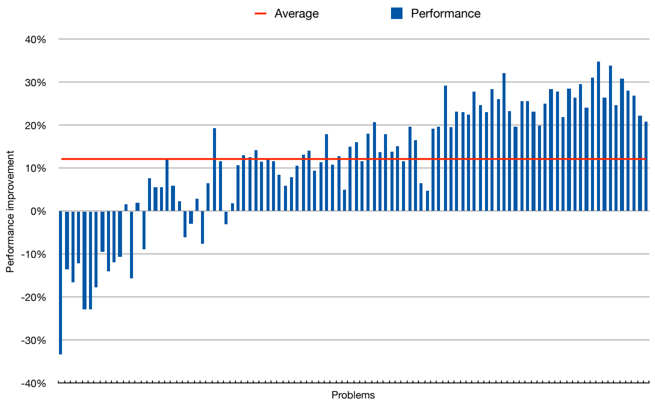


Figure 2. Performance improvement of EC4MAS on 100 problems with 300 nodes, *ec* = 0,02333, 4 colors.

1) Performance: First, we randomly picked a problem (*ref-problem*) among generated graphs, a genetic algorithm is used to get coupling values and to find the best exploring rate for Min Conflict with exploration (17,7% for *ref-problem*). We used this tuning to resolve all the generated problems. The figure V-B1-2 shows the performance improvement on all the 100 problems which consists in the difference between Min Conflict with exploration results and EC4MAS results. The problems are ordered by the number of cycles of the solving process. We can see that EC4MAS can generally improve the resolution on a set of problems with similar characteristics since the average improvement of EC4MAS is about 12,4%. There is a big difference of performance between easier problems (at the beginning) and more difficult ones (at the end). The *ref-problem* number of cycles for resolution with EC4MAS is 234, most of the problems with a negative improvement are under this value. The coupling used for *ref-problem* is a representation of the problem and gives specific informations on it. With problems much easier than that, the resolution is too specialized and instead of guiding the search process it introduces too much perturbations.

Resolution	Perf.	Tuning time	Efficiency
Min Conflict (17,7%)	100%	4	-
Optimal Min Conflict	124,71%	333	29,68%
EC4MAS (17,7%)	112,14%	22	231%

Table I
PERFORMANCE AND EFFICIENCY

In addition to the performance gain, we also focus on the tuning time of the system. Table I presents the performance gain of three different tuning and the efficiency of each method. The Min Conflict with 17,7% of exploration is taken as a reference for the measures. The tuning time is the sum of the time to find the optimal exploring rate for each problem for Optimal Min Conflict, and is the time to find the optimal exploring rate and the coupling values for *ref-problem* for EC4MAS. In the first case the tuning time is dependent on the number of problems and their hardness, in the second case only on the hardness of *ref-problem*. We can see here that the performance is much higher with optimal exploring rate, about 2 times more than EC4MAS, but the tuning time (in minutes) is about 15 times higher. In the end, the global efficiency (performance gain divided by tuning time) of EC4MAS is almost 8 times higher than Optimal Min Conflict. This shows that EC4MAS can learn the characteristics of a particular problem and is able to use this knowledge to really well solve similar problems with a limited tuning cost.

2) Genericity: ECM4AS uses the nodes degree to create spatial roles. EC4MAS has been developed to be as generic as possible. To illustrate the genericity we introduce here a new type of sensor for spatial role, the local clustering coefficient. This coefficient is based on triangles between neighbors of the node and the node itself. This coefficient is very useful to apprehend the difficulty of a graph coloring problem as seen in [13].

Spatial role	Perf. (cycle)	Improvement
Degree	234	11,5%
Clustering coefficient	223	17,04%

Table II
IMPROVEMENT OF DIFFERENT SPATIAL ROLES OVER MIN CONFLICT

Table II shows the performance on *ref-problem* with the Min Conflict with exploration, EC4MAS with degree and clustering coefficient for spatial role. We can see that the most specific sensor which is the clustering coefficient is more efficient than the others, about 17% than Min Conflict while the degrees are only 11,5% better than Min Conflict. EC4MAS could support several types of sensors for spatial roles, from more general one like degree to more specific one like clustering coefficient. EC4MAS is generic but is also dependent to the type of spatial role and sensor it uses.

VI. CONCLUSION AND FUTURE WORK

In this paper, we discuss the control problem in reactive self-organized systems. We distinguished two types of control, exogenous and endogenous. We describe the differences between them and pointed the main elements to address the creation of an endogenous control system, like observation, learning, dynamic and structural coupling. Then to illustrate the possibilities of an endogenous control system for problem solving, we considered the exploitation/exploration dilemma. We proposed a new model to create endogenous control in a self-organized multi-agent system, EC4MAS. This model is based on a social and a spatial organization and their coupling. These two organizations give informations on the current resolution strategy of the system and on its result, and the coupling allows the control system to dynamically adapt the strategy more efficiently.

EC4MAS is a generic model and could be used to solve different type of problems. The spatial organization could be adapted to use specific informations to let EC4MAS be able to solve different kind of problems. On more hard problems EC4MAS gives good improvements since the characteristics of the problem are used to better tackle it. EC4MAS makes the tuning of the system robust face to problem changes, the coupling of social and spatial organization gives pertinent solutions to encountered situations with a specific strategy. The tuning has not been changed when new problems are submitted. This is a great point because individual optimization is very expensive and could not be always used, more particularly when problems are dynamic. The endogenous self-organized characteristic of EC4MAS could efficiently limit the amount of time and resources to solve a problem.

Future work will focus on two main points, the dynamic adaptation of the coupling and a new social organization. The dynamic adaptation of the coupling will let the system find the good parameters during the resolution process. In addition social organization will be upgraded to give more specific informations on the current strategy. Instead of general information on the neighbors strategy, the real specific organization of a local situation should be considered.

REFERENCES

[1] F. Arlabosse, M.-P. Gleizes, and M. Occello, "Méthodes de conception de systèmes multi-agents." *Systèmes Multi-Agents - Collection ARAGO*, vol. 29, 2004.

[2] J. Ferber and O. Gutknecht, "A meta-model for the analysis and design of organizations in multi-agent systems," in *Multi Agent Systems, 1998. Proceedings. International Conference on.* IEEE, 2002, pp. 128–135.

[3] C. Bernon, V. Camps, M. Gleizes, and G. Picard, "Engineering adaptive multi-agent systems: The adelfe methodology," *Agent-oriented methodologies*, pp. 172–202, 2005.

[4] F. Armetta and S. Hassas, "Exploration expressiveness for self-organized systems : an application to the k-coloring problem," Laboratoire LIESP, Lyon, Tech. Rep., 2008.

[5] F. Klein, "Contrôle d'un sma réactif par modélisation et apprentissage de sa dynamique globale," Ph.D. dissertation, Université Nancy 2, 2009.

[6] M. Fehler, F. Klügl, and F. Puppe, "Techniques for analysis and calibration of multi-agent simulations," in *Engineering societies in the agents world V: 5th international workshop, ESAW 2004.* Springer Verlag, 2005, p. 305.

[7] B. Calvez and G. Hutzler, "Ant colony systems and the calibration of multi-agent simulations: a new approach," *MA4CS'07 Satellite Workshop of ECCS 2007*, 2007.

[8] I. Chades, B. Scherrer, and F. Charpillet, "A heuristic approach for solving decentralized-pomdp: Assessment on the pursuit problem," in *Proceedings of the 2002 ACM symposium on Applied computing.* ACM, 2002, p. 62.

[9] J. Campagne and A. Cardon, "Using morphology to analyse and control a multi-agent system, an example." in *STAIRS 2004: Proceedings of the Second Starting AI Researchers' Symposium.* Ios Pr Inc, 2004, p. 229.

[10] S. B. Thrun, "Efficient exploration in reinforcement learning," Computer Science Department, Pittsburgh, PA, Tech. Rep. CMU-CS-92-102, 1992.

[11] S. Wilson, "Explore/exploit strategies in autonomy," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*, 1996.

[12] A. Eiben, J. Van Der Hauw, and J. van Hemert, "Graph coloring with adaptive evolutionary algorithms," *Journal of Heuristics*, vol. 4, no. 1, pp. 25–46, 1998.

[13] A. A. Tsonis, K. L. Swanson, and G. Wang, "Estimating the clustering coefficient in scale-free networks on lattices with local spatial correlation structure," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 21, pp. 5287 – 5294, 2008.

Topology Control of Mobile Robot Networks for Optimal Coverage and Connectivity using Irregular Hexagonalization

Karthik Mahesh Varadarajan, Markus Vincze
ACIN, TU Vienna
Vienna, Austria
{kv,mv}@acin.tuwien.ac.at

Abstract— **A number of multi-robot networks (operating both indoors and outdoors) such as security and surveillance systems, industrial manufacturing, mobile defense robotic systems, mobile distributed monitoring systems as well as other cooperative robotic network systems like robotic soccer require multiple robots to provide maximal coverage and connectivity while being able to rapidly adapt to changes in environmental and channel conditions. Variation in link quality in the transition region is often a major concern in wireless multi-robot sensor networks. The quality of the wireless link varies temporally as well as spatially. Traditional algorithms attempt at solving this issue through a combination of deployment schemes and power control, leading to various topology control mechanisms. In the case of mobile robot networks, it is possible to resolve this issue using the additional degree of freedom rendered by the mobility of the robots. In this paper we introduce a novel robot node deployment scheme and topology control mechanism that aims at maximizing coverage, connectivity while providing reliable communication under varying network conditions. The topology control is achieved by modeling the inverse signal loss, multi-path fading and other effects and combining them to control the topology of the network in an irregular hexagonal static/ mobile multi-robot deployment scheme. Experimental data, together with simulation results demonstrate connectivity enhancement under signal loss with mobility based topology control.**

Keywords*- Multi-robot Networks, Topology Control, Outage, Fading Models, Hexagon, Deployment, Rayleigh, Log-Normal, Dynamic Network Topology, Mobile, Sensors*

I. INTRODUCTION

Efficient deployment of nodes in a multi-robot wireless sensor network is necessary to achieve good connectivity and reliability over time. A number of multi-robot networks such as security and surveillance systems, industrial manufacturing, mobile defense robotic systems, mobile distributed monitoring systems as well as other cooperative robotic network systems like robotic soccer require multiple robots to provide maximal coverage and connectivity while being able to rapidly adapt to changes in environmental and channel conditions. Numerous deployment strategies and topology control schemes have been developed for wireless sensor nodes over the last few years [23, 24, 25]. Schemes that involve both static and mobile nodes have been developed. These are directly applicable to mobile robot networks. Topology of multi-robot networks can also affected by the tasks assigned to the individual robots [27]. In addition, localization of mobile robots can be implicitly solved using the wireless sensor network framework [26]. Topological coverage management for mobile robots (especially with individual robots and cell divisions) [28] also contributes to the understanding of the problem of topology control for multi-robot networks.

The main criteria on which deployment schemes are currently developed are connectivity and coverage. In this paper, we focus on topology control of a mobile robot node network for 3-connectivity with blanket coverage. Terminology and algorithms from the domain of wireless sensor networks have been borrowed to this end.

The degradation of signal strength with the distance from the source node follows a logarithmic power law. The variation in the power level across time, in the case of a static wireless sensor network is largely dependent on changes in the environment parameters that determine the multipath fading effects. In the case of mobile networks, this also depends on other factors such as velocity of nodes, relative velocity of objects in the surrounding environment etc. The paper brings these effects into the modeling of the topology and hence the deployment scheme, thus improving the reliability and connectivity of the network. The base topology for the proposed deployment model is a hexagonal grid structure. The structure is designed by the superposition of static and mobile diamond topologies. Logarithmic models and various fading models have been combined for simulations. The metric for performance measurement is the '*Outage probability*' defined on the signal level falling below a threshold.

The central theme of this paper revolves around the development of a dynamic topology that is not just efficient in terms of reliability, connectivity and coverage but also has the capacity to alter its state in the case of an event (critical/non-critical) to obtain a new optimal state. The dynamic nature of signal strength variability in mobile robot networks is largely due to environmental changes, as these networks are deployed in areas where there is high channel variability and over which the user has little control. One such example setting is the equatorial rain forest, where it rains heavily in the afternoons and is extremely hot and sunny in the mornings. As a result, the fading model (rain fading [21]) of the environment changes largely, from time to time. These effects assume great significance in the case of surveillance, monitoring and defense robot networks. The fading models are important as they contribute to significant changes in the power level of the signal received by the nodes over short periods of time. The variation in the received signal at any point in the network depends on a number of factors like reflections, scattering etc. causing variations in the signal level in nano-sec to milli-sec range, a fraction of which is the time for measurement of the signal. Hence, it becomes necessary to adapt the network to the non-linearities in nature. One possible method would be to use power control. Another method would be to create a dynamic topology that adapts itself to changes in the environment.

Besides these non-linearities, there are numerous other effects that result in variation in the signal level received at a point in the network [12]. These include the fact that (1) the world is not flat but curved, (2) radio transmission areas are not circular, i.e. radios are not isotropic, (3) different radios have different ranges, (4) bi-direction links are not perfectly symmetric, (5) the quality of a link varies with time, (6) signal strength varies with distance in a complex way, (7) there are considerable fading effects and variations in fading effects. The goal of this paper is to design a dynamic topology that would enable to overcome these issues thereby creating a reliably connected network. While it is not possible to model each of these parameters individually, it should be possible to capture the group behavior approximately by fitting probability distributions.

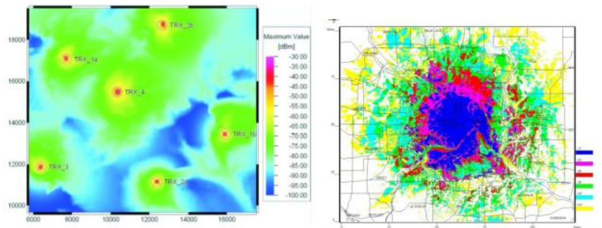


Figure 1. Typical radio power level contours with color representing the signal quality (Source: University of Stuttgart, Midwest Radio Association and [12]).

II. RELATED WORK

Node Deployments can be largely classified as structured and randomized. Using power control various network topologies have been achieved (MFCN, COMPOW, CRTC) [3]. Lifetime oriented, connectivity oriented and hybrid development schemes have also been analyzed [4]. While most of the sensor placement schemes are deterministic, stochastic node placement schemes are also being developed. One important node placement scheme uses the Power-Law placement [2] in which the degree of the nodes follows a power law. In the case of mobile nodes, schemes such as DSSA [5] and Incremental Self-Deployment algorithm [6] have been developed. An approach using the concept of potential fields repelling each other and stationary objects has been used in sensor node deployments [7] and for multi-robot networks using virtual angle forces for bi-connection in [29]. However, not all such approaches are feasible in an arbitrary environment. Few approaches utilize the power of mobile nodes in augmenting the quality of the network of static nodes. The theory developed in this paper corresponds to mobile nodes that have limited or constrained mobility (constraints could be task driven). The primary function of these nodes is to reposition themselves on the deployment grid in such a way that optimal link connectivity is obtained. Hexagonal node deployments have been expounded in [8, 9]. Efficiency of dynamic self-organizing networks has been demonstrated in [10].

Considerable work has been done in communication theory on the study of non-linearities in nature affecting radio propagation. While considerable amount of the theory from wireless communication has been borrowed into studying the radio propagation mechanisms of mobile wireless sensor networks, there are several differences. These stem from the fact that the sensors in general do not possess directional antennas and also the range of radios used on sensor networks is of the order of a few tens of meters to the maximum. Hence it is not possible to apply models such as the Clarke’s model directly without adaptation to wireless nodes. The variation of power level with distance, time and position (due to fading effects) has been observed to fit a log-normal distribution (Fig. 1). This is especially true in the case of indoor environments [14]. Furthermore, the arrival time of the multipath has been found to be Poisson and the log-mean value of amplitude decreases with increasing excess delay. Also, the path loss varies linearly with the delay spread [15]. A possible model for simulating the power values at adjacent points uses a superposition of bivariate Gaussians. Also another simulation model for urban radio propagation was obtained by Suzuki, called the Suzuki Distribution [16]. In the case of nodes in motion, the relative velocity causes a Doppler shift in frequency. This could also cause fading effects. However, this effect is excluded from analysis in the model developed in this paper. Mobile node localization based on this Doppler shift frequency is studied in [13].

III. DYNAMIC TOPOLOGY DESIGN

This paper concentrates on the developing a topology that captures the seven dynamic effects (referenced from [12]), which cause variation in the signal level received. The primary focus is on modeling the variation of the signal level with distance and in short intervals of time, characterized by fading models. These multipath effects also vary with the position of the receiver. Along with the other effects, the fading effects are captured implicitly and used to optimize node locations in the deployment scheme. The topology developed can be described in terms of the following structural topology features.

A. Hexagonal Base Topology

There are several advantages of using a hexagonal grid structure. Hexagonal structures offer the largest area coverage for a given number of points with the requirement of exact packing. Of all possible topologies, the least amount of energy is required for a random arrangement to be isomorphed into a hexagonal structure. Hence hexagonal topology is taken as the base topology for the network deployment scheme developed in this paper. Also, a hexagonal scheme covers the minimum overall perimeter for a given area, since it approximates a circle and at the same time allows for perfect packing. Hence links in a hexagonal network are closer to each other than in other schemes. This enables enhanced connectivity. Other advantages include: the lack of a blind spot, ease of deployment, adaptability to cell based sleep scheduling, ease in control of power level, ability to turn off transfer nodes to enable power savings, lack of network congestion due to simplicity of structure and simplicity of routing mechanisms [9]. Further, hexagonal networks give optimal performance in terms of minimal requirement of number of nodes [8].

B. Complementary Static-Mobile Nodes

Topology control is traditionally established using power control. However, the combination of mobile and static nodes provides a very powerful alternate deployment scheme. While it may be useful to have the entire network to be composed of mobile robot nodes, there are several practical considerations (task constraints) which prevent such a possibility for a non-critical network. Based on task constraints and desired work space of each robot, it can be expected that the mobile nodes do not move over a long range or continuously. Instead these mobile nodes are to alter their position in the grid only at periodic intervals.

C. Dynamic Hex Network

While a mobile robot network with full degree of freedom in mobility is highly adaptable to various topology requirements, the optimization is computationally extremely intensive for continuous real-time change in requirements and can lead to sub-optimal solutions with respect to connectivity of the network graph. Hence, a dynamic hex network which is completely connected (on an average) is employed as the underlying topology. However, due to time variation of fading and other effects, it is possible that the network becomes disconnected. It is this probability that the algorithm developed in the paper tries to minimize. This is called the outage probability. In other words, the no-outage probability (i.e. 1 - outage probability) is to be maximized.

Such a dynamic hex network formed by superimposing a diamond topology of static nodes on a diamond topology of mobile nodes is depicted in Fig. 3. Fig. 2 illustrates the constraints on the optimization of position of a mobile node. It can be seen that the position of a mobile node can vary within a triangular area bounded by the neighboring nodes. For simplicity, the position of the nodes is assumed to vary linearly from or to the neighboring nodes in this triangular region.

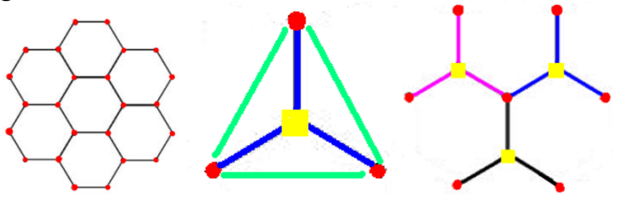


Figure 2. Left: Hexagonal grid topology, Center: Each mobile node is connected to three other nodes in a triangular area of constrained motion. Right: Unique colors represent the link minimization criteria for each node. Clearly, the optimization of one node does not affect the other nodes.

Such a topology preserves the static power relationships. Thus, it can be seen that each node can optimize its position dynamically, without regard to the other nodes. Typically the time interval for optimizing can be set based on the frequency of occurrence of the natural event. The optimization process can occur by decentralized processing. Also, all mobile nodes can reposition themselves simultaneously. Since the static nodes do not require to reposition themselves or identify the position of the mobile nodes, they can be devoid of memory and information about the network topology. On the other hand, the mobile nodes require knowledge of the position of the static nodes and also should be capable of localizing themselves so that they move and relocate to an optimal position.

D. Non-Ideal Environment Modeling

If the network is ideal, then all the links between the nodes will have equal weight, i.e., the radio range will be isotropic. For the ideal case the power topology is composed of regular hexagons when the deployment topology is made of regular hexagons (Fig. 3).

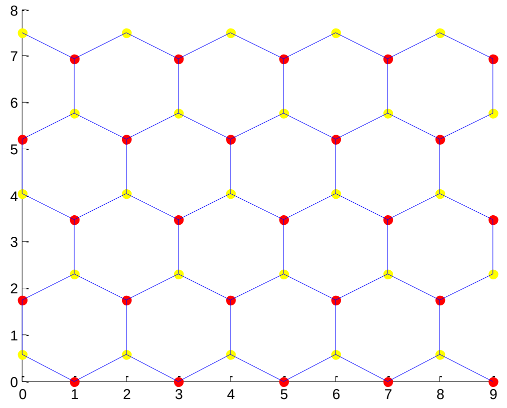


Figure 3. Array of superimposed static (red) and mobile (yellow) diamond topologies forming the Hexagonal Dynamic Topology. It can be seen that the mobile and the static nodes fall on alternate layers. The blue segment represent ideal links

In a practical scenario, the major large scale path loss in the signal received from a radio can be attributed to reflections, scattering and diffraction. The effect of these factors can be modeled individually. An alternate method would be to model the path loss in a single term as log-normal. A Gaussian factor is included in the log-normal model to indicate the variation in the values of the received power at different points at the same distance from the transmitter [17].

$$PL(d) = \overline{PL\ (d_0)} + 10nlog\frac{d}{d_0} + X_\sigma$$

All the power values in the above equation are represented in terms of dB. $PL(d)$ is the Path Loss at a distance d from the transmitter. X_σ is the zero-mean Gaussian distributed random variable (in dB) with σ (also in dB) as variance. n is the Path Loss exponent and $\overline{PL\ (d_0)}$ represents the average power level at a reference distance d_0 from the transmitter. The received power is related to the power loss as below.

$$P_r(d) = P_t(d) - PL(d)$$

Here, $P_r(d)$ *and* $P_t(d)$ are given dBm and PL (d) is given in dB. The value of the path-loss exponent varies typically from 1 to 6 depending on the availability of a line of sight path. It is higher for the case of larger spaces or spaces with more obstructions and in places where there is no line of sight. A good way to represent numerically the probability that the signal level will exceed a particular threshold is given by the Q function (z is the random variable representing the power level)[17].

$$Q(z) = \frac{1}{\sqrt{2\pi}} \int_{z}^{\infty} \exp(-\frac{x^2}{2}) dx$$
$$= \frac{1}{2}(1 - \text{erf}\left(\frac{z}{\sqrt{2}}\right))$$

where $Q(z) = 1 - Q(-z)$

The probability that the received signal (in dB power units) will exceed a certain value γ can be calculated from the cumulative density function of the log-normal distribution as

$$\Pr[P_r(d) > \gamma] = Q(\frac{\gamma - \overline{\overline{P_r(d)}}}{\sigma})$$

Thus values from the Cumulative Distribution Function (CDF) of the function gives the probability that the signal power level is above a certain threshold. This is the metric used in this paper for the optimization. This metric is directly related to the outage probability (which uses the signal amplitude value instead of the power value) and hence increase in the probability of the above function is equivalent to increasing the no-outage probability.

Propagation models for outdoor and indoor wireless communication include the Longley Rice Model, Durkin's Model, Okumura Model, Hata Model, Lee Model, Walfisch and Bertoni Model, Attenuation Factor Model and the Partition Losses Model [17]. Further, there are small scale fading and multipath effects causing variations in the received signal level. In the case at hand, the multipath fading occurs only due to rapid changes in the signal strength over a small travel distance and time dispersion caused by multipath propagation delays. The spatial variations can be essentially modeled as variations across time as the mobile robot has to change its position through time in order to reach its new position. Multipath propagation effects are largely due to the reflections of the signal by the environmental features resulting in multiple paths for the signal arrival at a point and these are delayed or phase shifted. This phenomenon is called Multipath time delay spread. Thus, fading due to shadowing and other location specific properties is called 'Slow Fading', whereas that due to multipath forms 'Fast Fading' (Fig. 4). Here two possible cases arise. One possibility is that the frequency response of the channel is constant or flat and the other possibility is that it varies with the frequency. Correspondingly we have Flat Fading and Frequency Selective Fading. Besides, small scale fading based on Doppler spread can also create fading effects. Again we have Fast and Slow fading corresponding to Doppler Spread based fading. The only fading of concern in this paper is the flat fading based on the Multipath time delay spread. This is because the system feature response can be assumed to be flat and doesn't change with the frequency. Also, since the mobile robots are not required to measure power when in motion, Doppler effects are not of concern. The time range of concern is of the order of a few milliseconds.

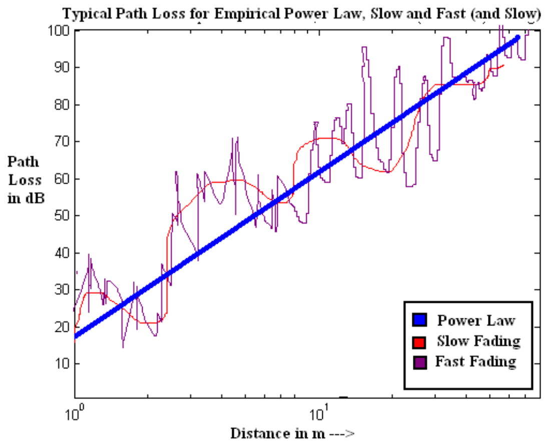


Figure. 4. Typical Path Loss with Power Law, Slow Fading added and Fast Fading added.

While the power loss in the case of slow fading (flat) is represented conveniently by the log-normal distribution, the total power received (after the power loss) in the case of fast fading can be represented in the form of other statistical distributions [20, 22] such as the Rayleigh, Rician, Nakagami-m, Weibull, Lognormal, Suzuki, Gauss-Markov or HMM distributions. In this paper, the slow flat fading using log-normal distribution and fast flat fading using Rayleigh have been used. By fitting measured data, it is possible to calculate the parameters of the distributions. Once the parameters are calculated, it is relatively easy to calculate the integral of the Cumulative Distribution Function above a certain threshold. This gives the probability that the signal power was above a certain threshold.

Rayleigh Distribution:

$$p(r) = \left\{\frac{r}{\sigma^2} \exp\left(-\frac{r^2}{2\sigma^2}\right)\right\} \; 0 \le r \le \infty$$
$$p(r) = 0 \;\; r < 0$$

where σ is the RMS value of the received signal voltage and r is the envelope of the signal.

Rician Distribution:

$$p(r) = \left\{\frac{r}{\sigma^2} \exp\left(-\frac{r^2 + A^2}{2\sigma^2}\right) I_0(\frac{Ar}{\sigma^2})\right\} \; A \ge 0 \; r \ge 0$$
$$p(r) = 0 \;\; r < 0$$

'A' refers to the mean component adding to the signal. Rayleigh distribution is used where most components are scattering or multipath components. Rician distribution is used when there is a strong Line of Sight (LoS) component.

Another method of optimizing the position of the mobile robots would be to obtain the PRR (Packet Reception Rate) and use it as the metric for comparison. This is natural to use as it would take into account the modulation effects and hence the frequency selectiveness of the fading, irrespective of whether it is narrowband or wideband. However, since the PRR variation is very steep in the sigmoidal region (the transition region) it is difficult to use it as a metric. Further, the PRR depends on the digital modulation scheme employed [18]. In order to simplify the working of the algorithm, the power levels are considered directly, yielding the outage probability. A further simplification would be to use the RSSI values directly instead of calculating the RF power levels in dB. This is acceptable as the relationship between the RSSI and the RF power levels is a linear shift, and this can be modeled by an equivalent shift in the log domain [19].

E. Algorithm

Given the constraints on the mobile robot nodes, the following heuristic algorithm is proposed in order to optimize the positioning of the mobile node.

Training Mode

i) The hexagonal deployment of the mobile and static nodes is carried out.
ii) An efficient hill climbing approach can be implemented to relocate the mobile node in each triangular cell at its optimal position. But, this is computationally demanding on the mobile node. An easier approach is to create power level bins and estimate the optimal position by a brute force approach. For the case of pure slow fading (log-normal shadowing), an estimate of the optimal position can be computed.
iii) The mobile nodes walk through a triangular region bounded by the static nodes gathering data on the power levels from nearby nodes at preassigned bin locations.
iv) A Maximum Likelihood Estimation (MLE) is performed on data obtained to estimate the distribution parameters.
v) The outage probability is estimated from the distribution model.

Redeployment Strategy

i) An initial value of desired outage probability is chosen.
ii) The mobile robots, based on the data collected during the training mode calculate the position where the no-outage probability from each static node is satisfied.
iii) If the mobile node is successful in finding a bin/point, the value of the desired no-outage probability is increased and step ii repeated.
iv) The process is terminated when the mobile node does not find a bin satisfying the no-outage probability criterion for all three nodes.
v) The mobile node moves to the last successful position identified.

IV. EXPERIMENTS, SIMULATIONS AND RESULTS

Experiments were conducted to measure the power levels at various distances (power bins). The chosen distances were 1m, 5m, 10m, 20m and 25m. The RSSI values obtained were used in the fitting of distributions. The setting of the experimental process was in a parking lot, an environment with both fast and slow fading effects. Multiple measurements in short time intervals yield the multipath fading components. In our testing, we simulate the environment for multiple mobile node using parameters obtained from the experimentation on a few nodes. Parameters of the distributions were obtained for Log-Normal, Weibull and Rayleigh distributions.

TABLE I

Distance	RSSI in –dB for 25 positions				
1	38.1140	37.6980	37.5220	33.7150	31.9310
	31.6930	31.6880	31.6250	31.3300	31.1170
	30.6310	30.6100	30.5630	30.5120	30.5120
	30.2220	29.9970	29.5960	29.2120	29.2100
5	44.7210	44.6730	44.5830	44.3890	44.3600
	44.1220	44.1000	43.6280	42.2310	41.7800
	41.6220	41.2120	41.0000	40.9180	40.8820
	40.5100	39.6210	39.2420	38.1510	37.2830
10	46.2310	46.2060	45.2330	44.6670	44.5460
	44.5410	44.5320	44.4440	44.4270	44.2120
	44.1970	44.1660	43.2220	42.7700	42.4960
	41.5350	41.5260	41.1230	41.1130	40.3230
20	49.8330	49.6580	49.3610	49.3580	49.1380
	49.1110	48.9440	48.1450	47.4750	47.3630
	47.3450	47.2530	47.2330	47.1550	47.1230
	47.0030	46.9430	46.7350	46.6330	46.3330
25	50.1720	49.7220	49.5570	49.4880	49.2880
	49.2220	48.9200	48.8480	48.5330	48.2580
	47.3900	46.7520	46.2370	46.0020	45.7370
	45.5870	45.4790	45.2570	45.2360	45.2290

Table 1. Distance in meters and Average RSSI values in –dB for 25 positions.

A Linear Regression fit for the experimental data yielded values of n and X_σ (for the Log-Normal Fading model) as 1.1419 and 4.085. Correspondingly, the following approximate parametric values were obtained for the log-normal fading Param1 (μ) = [3 2 1 0.6 0.5] over the r distance bins and Param2 (σ) = [1 0.7 0.5 0.25 0.2]. For Rayleigh, values obtained were (σ) = [25 12 8 2 1] and that for Weibull (μ) = [30 10 4 2 1.5] and (ω) = [2 2 2 6 4]

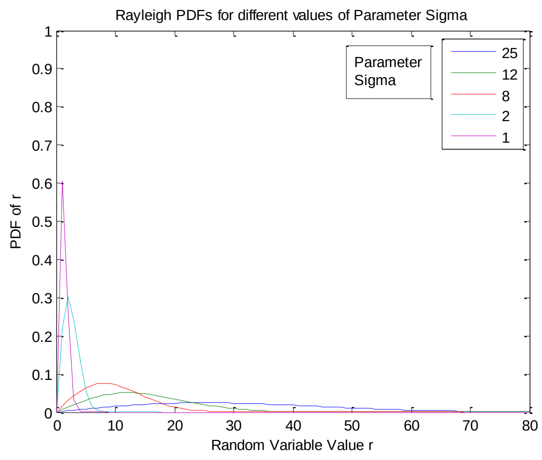


Figure. 5. Rayleigh distributions for different values of parameters in inset.

The random variable 'r' represents the values the distribution can take. It is directly related to the power values measured. In Fig. 5, the different curves indicate the Rayleigh distributions for different values of parameters. The parameters are obtained at different distances of the mobile node from the static node. The higher the distance, the larger is the value of the Rayleigh parameter. It is seen that curves corresponding to larger distances are flatter. Thus, it can be seen that towards larger values of r, the area under the curve beyond r (and hence the no-outage probability) is larger than for smaller parametric values or shorter inter-node distances. Since this variation is non-linear and differs from one static node to the other, it is possible to find an optimal position for the mobile node by brute force.

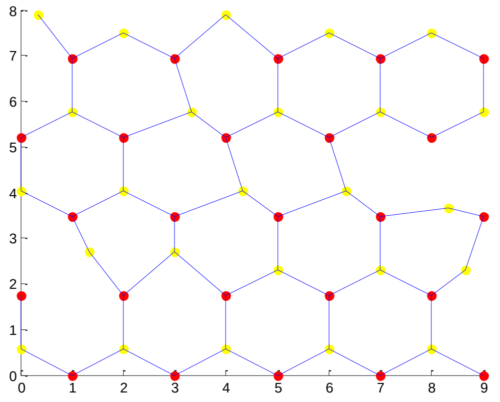


Figure. 6. Graph showing connectivity pattern after the node repositioning (based on the connectivity of the original regular hexagonal topology)

Using the heuristic node repositioning algorithm operating on the experimental data in Table 1, the overall average outage probability of the simulated network, using the Rayleigh distribution was increased from 0.60 to 0.80 (assuming 15m as the initial hypothetical location of each mobile node from the other three static nodes). Since the network is designed to choose dynamically thresholds in order to increase the no-outage probability, the result is on expected lines. Also, in order to keep the algorithm fast, only few bins were used (5*5*5 bins one each at 25, 20, 10, 5 and 1 meter from the static nodes). The thresholds were increased in coarse steps of 0.05. The results can be enhanced by proper choice of step size and distribution based on the environment at hand.

V. Conclusion And Future Work

In this paper, we have presented a novel mobility based topology control scheme for mobile robot networks. The method presented uses a combination of static and mobile nodes in an irregular hexagonal framework combined with dynamic topology management to enhance reliability and connectivity. Experiments and simulations demonstrate reduced outage probability.

The current algorithm uses a brute force optimization approach. Investigation of alternate approaches to identify the optimal location form future work. While localization using the Log-Normal fading model is easy, it does not take into account the fast fading effects. These can be modeled using Rayleigh or Rician fading, but with increased computational requirements. One way of achieving this would be to use a recursively updating procedure to find the optimal location. It could be designed such that at each step of the process, the node improves its positioning and the process converges in a limited number of steps. Further, the radio model and effects of the modulation scheme can be included in the modeling as well. However, in this case, it is necessary to alter the fading model as well. This is because, if the implementation is not narrowband but instead uses a wideband channel, then the effects of frequency selective fading come into play. Alternate schemes that enable a mobile node to find the optimal position by moving towards it will necessitate changes to the fading model by introducing effects of Doppler spread. Also, a PRR based measure will be the most suitable for optimization. Future work will involve incorporating the above in the modeling.

References

[1] G. Hoblos, M. Staroswiecki, and A. Aitouche, “Optimal Design of Fault Tolerant Sensor Networks,” ICCA, Sept. 2000, pp. 462–472.
[2] M. Ishizuka, M. Aida, “Achieving Power-law Placement in Wireless Sensor Networks”, ISADS 2005.
[3] Bhaskar Krishnamchari, “Networking Wireless Sensor Networks”, Edition 1, Cambridge 2005.
[4] K. Xu, H. Hassanein, G. Takahara, Q. Wang, “Relay Node Deployment Strategies in Heterogeneous Wireless Sensor Networks: Single-Hop Communication Case”, IEEE GlobeCom 2005.
[5] N. Heo, Varshney, P.K., “An intelligent deployment and clustering algorithm for a distributed mobile sensor network”, IEEE International Conference on Systems, Man and Cybernetics, 2003.
[6] A. Howard, M. J. Mataric, G. Sukhatme, “An incremental self-deployment algorithm for mobile nets”, J. Autonomous Robots, 2003.
[7] A. Howard, M. J. Mataric, G. Sukhatme, “Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem”, DARS, 2002.
[8] M. M. Iqbal, I. Gondal, L. Dooley, “Dynamic Symmetrical Topology Models for Pervasive Sensor Networks”, INMIC 2004.
[9] H. Tian, H. Shen, “An Optimal Coverage Scheme for Wireless Sensor Network”, ICN 2005.
[10] S. S. Doumit, D. P. Agrawal, “Self-Organized Criticality & Stochastic Learning Based Intrusion Detection System for Wireless Sensor Networks”, Military Communications Conference, 2003.
[11] D.W. Gage. “Command control for many-robot systems”, AUVS-92.
[12] D. Kotz, C. Newport, C. Elliott, “The mistaken axioms of wireless-network research”, Dartmouth College Computer Science Technical Report TR2003-467, July 18, 2003.
[13] A. Lakhzouri, E. S. Lohan, R. Hamila, M. Renfors, “Extended Kalman Filter Channel Estimation for Line-of-Sight Detection in WCDMA Mobile Positioning”, EURASIP 2003.
[14] H. Hashemi, “Impulse Response Modeling of Indoor Radio Propagation Channels”, IEEE J. on Comm. 1993, Vol.11, pp.967-978.
[15] H. Nikookar, H. Hashemi, “Statistical Modeling Of Signal Amplitude Fading Of Indoor Radio Propagation Channels”, IEEE UPC 1993.
[16] H. Suzwi, “A Statistical Model for Urban Radio Propagation”, IEEE Trans. on Communications 1977, Vol. 25, pp. 673 - 680.
[17] T. S. Rappaport, “Wireless Communications: Principles and Practice”, Edition 1, Prentice Hall, 1996.
[18] M. Zuniga, B. Krishnamachari, “Analyzing the Transitional Region in Low Power Wireless Links”, IEEE SECON 2004.
[19] TMOTE SKY Manual.
[20] JPL’s Wireless Communication Reference Website: http://wireless.per.nl/reference/about.htm (accessed 15 Jan 2011)
[21] “Uncorrelated Rain Fading and its Impact on Frequency Re-Use and Antenna” RPE, Specifications, IEEE 802.16 BWAG.
[22] Joel T. Johnson, “Computer Simulations of Rough Surface Scattering”, Nanostructure science and technology, 2007, pp.181-210.
[23] Iino, Y., Hatanaka, T., Fujita, M., “Dynamic topology optimization for dependable sensor networks”, IEEE CCA 2010.
[24] J.M. Hong, Q. Zhang, L.M.Ni, “Energy-efficient opportunistic topology control in wireless sensor networks”, MobiOpp 2007.
[25] J. Pan, Y.T. Hou, L. Cai, Y. Shi, S.X. Shen, “Topology control for wireless sensor networks”, IEEE MobiCom 2003.
[26] Maxim A. Batalin, Gaurav S. Sukhatme, Myron Hattig, “Mobile Robot Navigation using a Sensor Network”, IEEE ICRA 2004.
[27] Ceng Xian-yi, Li Shu-qin, & Xia De-shen, “Study of Self-Organization Model of Multiple Mobile Robot”, IJARS 2005.
[28] Sylvia C. Wong, Bruce A. MacDonald, “A topological coverage algorithm for mobile robots”, IEEE IROS 2003.
[29] Casteigts, A., et al., “Biconnecting a Network of Mobile Robots Using Virtual Angular Forces”, IEEE VTC, 2010.

UAV Speed Estimation With Multi-bit Quantizer in Adaptive Power Control

Hyeon-Cheol Lee
Smart UAV Development Center
Korea Aerospace Research Institute
115 Gwahangno, Yuseong-gu, Daejeon, Rep. of Korea
Email: hlee@kari.re.kr

***Abstract*—The adaptive power control with multi-bit quantizer of Code Division Multiple Access (CDMA) systems for communications between multiple Unmanned Aerial Vehicles (UAVs) with a link-budget based Signal-to-Interference Ratio (SIR) estimate which has distance information is applied to three inner loop power control algorithms. The speed estimation performance of these algorithms with their consecutive Transmit-Power-Control (TPC) ratios are compared to each other, and it is concluded that the speed can be estimated and shows full linearity with the TPC ratio information of Consecutive TPC Ratio Step-size Closed Loop Power Control (CS-CLPC) and Fixed Step-size Power Control (FSPC).**

***Keywords*-speed estimation; adaptive power control; link-budget; SIR; multi-bit quantizer**

I. INTRODUCTION

The communication system of an Unmanned Aerial Vehicle (UAV) with multiple UAVs requires a mobile wireless network to share data between UAVs. One communication network protocol that may be used is Code Division Multiple Access (CDMA). CDMA differs from both Frequency Division Multiple Access (FDMA) and Time Division Multiple Access (TDMA) in that it uses the same frequency for multiple users.

Since all users utilize a single frequency, the signal from each UAV may interfere with other UAVs' receivers [1]. This is referred to as the near-far effect. To eliminate the near-far effect in CDMA systems, the transmission signal power from every UAV must be the same as the signal power at the receiver. This technique of controlling the magnitude of the transmission power according to the distance between the UAV and the Base Station (BS) is officially termed power control. It equalizes the received power and eliminates the near-far effect, though it is subject to such complications as path loss, shadowing, multi-path fading, etc.

This power control technique is differentiated into open loop power control and closed loop power control. The closed loop power control is further divided into inner loop power control and outer loop power control. The inner loop power control is responsible for adjusting the power transmitted to maintain the received Signal-to-Interference Ratio (SIR) at the BS at a level equal to that at the SIR_{target}. The outer loop power control is responsible for setting the SIR_{target} based on the Bit Error Rate (BER) or service requirement.

In general, an aircraft measures its air speed by pitot tube and calculates its ground speed by Global Position System/Inertial Navigation System (GPS/INS). In these days, there is an effort that optic flow measurements from CCD camera have been used to augment GPS/INS to provide more precise velocity information [2]. In this paper, the speed is calculated with SIR estimates. Conventional SIR estimates [3] consider only the transmission power and the link-gain, but this paper takes into account the link-budget, which has more realistic parameters including distance information than the link-gain. Using the SIR estimate that reflects the link-budget, speed estimation [4] is introduced based on a Consecutive Transmit-Power-Control Ratio (CTR) [5]. The proposed speed estimation method is applied to three algorithms with one, two, and three-bit level quantizers, and the results are compared.

This paper is organized as follows: The literature related to this work is surveyed in Section II. The inner loop power control is described in Section III. The concept of the link-budget based SIR estimate is introduced in Section IV, followed by description of simulation environments in Section V. Section VI gives details of the speed estimation. The simulation results are analyzed in Section VII. Finally, conclusions are drawn in Section VIII.

II. RELATED WORK

Kim, Lee, and Kim introduced in [6] the Adaptive Step-size Closed Loop Power Control (AS-CLPC) algorithm for a narrowband CDMA system. This algorithm adapts its power control step-size based on the optimal factors determined with the mean fade duration which is inversely proportional to the maximum Doppler frequency. Nourizadeh, Taaghol, and Tafazolli [7] proposed the Blind Adaptive Closed Loop Power Control (BA-CLPC) in which the power control step-size is adjusted to cope with the user mobility. Taaghol [8] introduced the Speed Adaptive Closed Loop Power Control (SA-CLPC) algorithm in which the power control step-size is selected based on the user speed estimation categorized by speed ranges. Lee and Cho [9] proposed the Mobility Based Adaptive Closed Loop Power Control (M-ACLPC) algorithm in which the power control step-size is adjusted depending on the combination of the cumulative information of the three power control commands and speed estimation.

Patachaianand and Sandrasegaran [5] compared performances of the AS-CLPC, BA-CLPC, SA-CLPC, and M-ACLPC in terms of Power Control Error (PCE) under the same simulation environment. In their comparisons, the AS-CLPC showed the best performance when the target speed was lower than 25km/h, while the SA-CLPC was the best when the speed was greater than 25km/h.

Patachaianand and Sandrasegaran [5] presented the Consecutive TPC Ratio (CTR) Step-size Closed Loop Power Control (CS-CLPC) algorithm whose power control step-size is determined based on a parameter called CTR. They measured the moving target speed by CTR, then, calculated the PCE as the Root Mean Square (RMS) of the difference between the received SIR and the SIR_{target}. They also suggested in [4] the mapping equation and mapping table which can yield accurate speed estimation using CTR.

III. Inner loop power control

In CDMA, the process of inner loop power control occurs as follows: In the reverse link direction (from the UAV to the BS), the transmission power information goes to the BS. At the BS, the SIR_{target} and the received SIR are calculated from the transmission power, the link-gain, and the noise power. Based on these factors, the BS sends a Transmit-Power-Control (TPC) command to each UAV at rate of 1500Hz, or Sample Time (T_S) (= 0.667ms) in the forward link direction (from the BS to the UAV). This power equalization increases the maximum communication number between UAVs and consequently eliminates the near-far effect. These procedures [3][5] are represented in (1) and (3).

$$P_i(t+1) = P_i(t) + \delta_i(t) \cdot TPC_i(t) \tag{1}$$

where $P_i(t)$ is the transmission power, $\delta_i(t)$ is the power control step-size, and $TPC_i(t)$ is the TPC command for the ith UAV at time t.

$$x_i(t) = SIR_{target,i}(t) - SIR_i(t) \tag{2}$$

where $SIR_{target,i}(t)$ (=$SIR_{target}(t)$ here) is the target SIR and $SIR_i(t)$ is the received SIR for the ith UAV at time t.

Non-uniform quantizer [10] used in voice coding is introduced that (m+1)-bit TPC is adopted $TPC_i(t) = \{C_0C_1 \cdots C_m\}$ where C_0 is the sign bit.

$$\begin{aligned} &TPC_i(t) = sign(x_i(t)) \cdot \\ &\lfloor \tfrac{log((1+\mu|x_i(t)|)/R_P)}{(1+\mu)}(2^m - 1) \rfloor \end{aligned} \tag{3}$$

where $\mu = 2^{(m+1)}$-1 and R_P is dynamic range of power adjustment.

Table I
TPC Coefficients

m	d	multilevel	R_P
1	0.5	± 1,2	0.1
2	0.25	± 1,2,3,4	0.3
3	0.125	± 1,2,3,4,5,6,7,8	0.5

IV. Link-budget based SIR

The conventional SIR estimate of the ith UAV in CDMA is described as follows :

$$SIR_i(t) = \frac{P_i(t)G_i(t)}{\sum_{j \neq i} P_j(t)G_{ji}(t) + P_N} \tag{4}$$

where $G_i(t)$ is the link-gain between the ith UAV and the connected BS, and $P_i(t)$ is the transmission power from the ith UAV. $G_{ji}(t)$ is the link-gain between the jth UAV and the BS to which the ith UAV connects. Equation (4), however does not have distance information, therefore, $SIR_i(t)$ can not be measured by distance step.

This paper introduces the link-budget based SIR as

$$SIR_i(t) = \frac{P_{R,i}(t)}{\sum_{j \neq i} P_{R,ji}(t) + P_N} \tag{5}$$

where $P_{R,i}$ is the received power from the ith UAV. $P_{R,ji}$ is the received power from the jth UAV with the BS to which the ith UAV connects. The received power is affected by factors including the free space loss [11] which has distance information and gaseous path loss [12] varied by humidity. Speed estimation can be possible with distance per T_S and $SIR_i(t)$ is measured with $P_{R,i}$ by distance variation.

The power delivered to the receiver [11] is :

$$P_{R,i} = G_{T,i} \cdot P_{T,i} \cdot G_{R,i}/(L_F(D_i) \cdot L_G(D_i)) \tag{6}$$

where $G_{T,i}$, $P_{T,i}$, $G_{R,i}$, $L_F(D_i)$, and $L_G(D_i)$ are the transmission antenna gain, the transmission power, the receive antenna gain of the ith UAV, the free space loss, and the gaseous path loss, respectively (the component loss is ignored here.). The speed is estimated from moving distance of free space loss which has distance information. D_i is the distance between the ith UAV and the BS in kilometers.

$$L_F(D_i)(dB) = 92.44 + 20log_{10}(F) + 20log_{10}(D_i). \tag{7}$$

F is the frequency in gigahertz and the specific attenuation due to dry air and water vapor from sea level to an altitude of 5km can be estimated by (8).

$$L_G(D_i) = (\gamma_0 + \gamma_W) \cdot D_i. \tag{8}$$

P.676-5 of [12] shows equations of γ_0 (attenuation for dry air) and γ_W (attenuation for water vapor). These attenuations are dependent on σ, the water vapor density (g/m^3) specified in Table II from P.836-3 of [12]. The bigger the σ is, the larger the attenuation is.

Table II
WATER VAPOR DENSITY AT DIFFERENT SEASONS AND REGIONS

	Jan.	April	July	Oct.
Coast (edge of continent)	5	10	20	10
Inland (inside continent)	5	5	10	5
Ocean	20	20	20	20

The noise power [11] is :

$$P_N = k \cdot T \cdot B \tag{9}$$

where k is the Boltzmann constant (1.38×10^{-23}J/K), T is the temperature in Kelvin, and B is the equivalent bandwidth in hertz.

V. Simulation environments

This section presents a simulation of the speed estimation using (5). The frequency, the temperature, the pressure, the water vapor density, and the bandwidth are set to 2.0GHz, 288K, 1013hPa, 20 (Summer in Coast), and 5MHz, respectively. In Figure 1, five UAVs are arranged and D_is are set to 250m, 500m, 750m, 1000m, and 1250m. The antenna gain of each UAV is set to 0dB, as is the antenna gain of the BS. UAV1 to UAV5 complete their power control by FSPC so that each transmission power shown in Table III is different. Then, UAV1, which is 1250m away from the BS, starts to move outward. It moves at five different speeds and measures the CTR at each speed. SIR_{target} is set to the transmission power of UAV3 on this simulation. This simulation applies one UAV movement and outward/inward direction only.

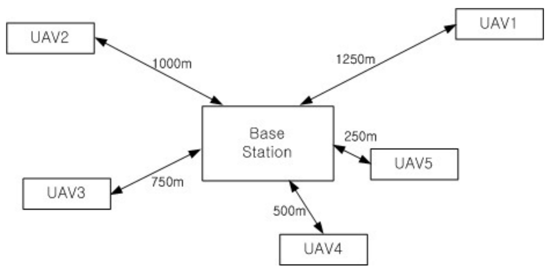


Figure 1. Simulation formation

VI. Speed estimation

There are several algorithms addressing inner loop power control, including CTR Step-size Closed Loop Power Control (CS-CLPC) [5], Adaptive Step-size Closed Loop Power Control (AS-CLPC) [6], Fixed Step-size Power Control (FSPC), etc.

Table III
INITIAL CONDITION OF UAV1 TO UAV5

	P_T (dB)	G_T (dB)	G_R (dB)	D_i (m)
UAV1	+8.048	0.0	0.0	1250
UAV2	+4.314	0.0	0.0	1000
UAV3	+0.000	0.0	0.0	750
UAV4	-5.229	0.0	0.0	500
UAV5	-13.979	0.0	0.0	250

This section investigates changes in transmission power for the above three algorithms with the link-budget based SIR. UAV1 moves outward for 42000*0.667ms (the number of the sample = 42000) at the six different speeds listed in Table IV; 100km/h, 200km/h, 300km/h, 400km/h, 500km/h, and 600km/h. As UAV1 moves away, the three inner loop power control algorithms alter the transmission power to compensate for the distance between BS and UAV1. A faster mobile is likely to receive more consecutive TPC command than a slower one. Therefore, average of CTR on certain period can be used to reflect user speed. Equation (10) [5] measures the CTR as follows:

Table IV
UAV1 MOVING DISTANCE FOR 28.0 SEC (=42000*0.667ms).

speed (km/h)	speed (m/s)	moving distance (m)
100	27.7778	777.78
200	55.5556	1555.56
300	83.3333	2333.33
400	111.1111	3111.11
500	138.8889	3888.89
600	166.6667	4666.67

$$CTR(t) = \sum_{n=t-m+1}^{t} \frac{d \cdot \mid TPC(n) + TPC(n-1) \mid}{m} \tag{10}$$

where d is a scale factor (see Table I), $m = t$ if $t < w$, and $m = w$ if $t \geq w$. w is the maximum size of the window average.

A. CS-CLPC

Patachaianand and Sandrasegaran [5] introduced the CS-CLPC algorithm, where the step-size is adjusted as shown in (11).

$$\delta(t) = \frac{\alpha}{1 - \beta \cdot min\{CTR(t), CTR_{max}\}} \tag{11}$$

where α, β, and CTR_{max} are constants.

B. AS-CLPC

Kim, Lee, and Kim [6] suggested the AS-CLPC algorithm. This algorithm adapts its step-size based on TPC history. The step-size is given by (12).

$$\delta(t) = \begin{cases} \delta(t) \cdot K, & TPC(t) = TPC(t-1) \\ \delta(t)/L, & Otherwise \end{cases} \tag{12}$$

where K and L are positive real constants with ranges of $1 < K$ and $1 < L < 2$.

C. FSPC

In this simulation, the algorithm uses a fixed step-size.

VII. Simulation results

Six different speeds are measured with CTR, and the relationships are shown in Figure 2 to Figure 7 according to two window sizes and three quantizers. The CS-CLPC and FSPC algorithms show a linear relationship between speed and CTR, in addition they show more linearity and can estimate higher speed as quantizing bits are increased. Therefore, using the CS-CLPC or FSPC, the speed of the vehicle can be measured by mapping the information from CTR. The AS-CLPC algorithm, however, deviates from linearity. In addition, the same results are obtained with different window sizes.

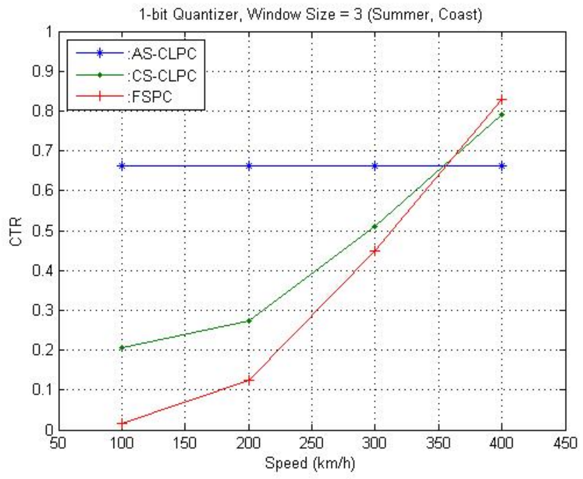


Figure 2. CTR vs. UAV1 speed at 1-bit Quantizer, window size 3 (Summer, Coast)

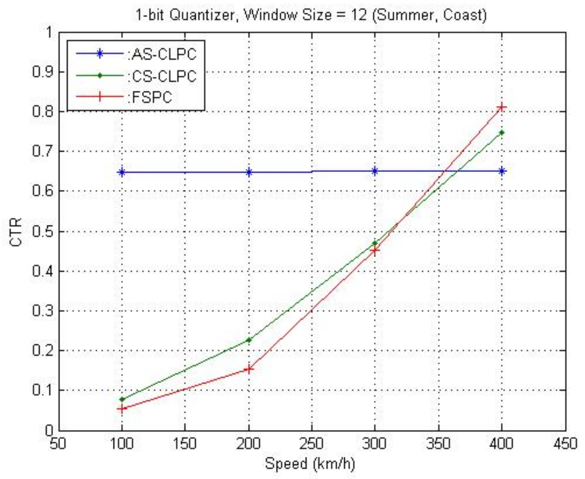


Figure 3. CTR vs. UAV1 speed at 1-bit Quantizer, window size 12 (Summer, Coast)

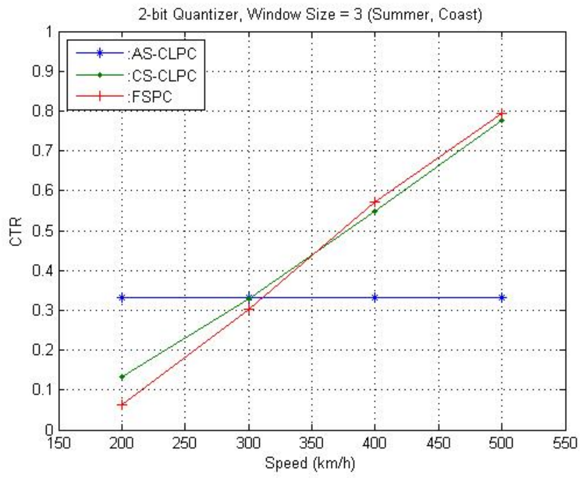


Figure 4. CTR vs. UAV1 speed at 2-bit Quantizer, window size 3 (Summer, Coast)

VIII. Conclusion and Future Work

This paper introduced a speed estimation of three different inner loop power control with consecutive TPC ratios which use the link-budget based SIR in the CDMA communication systems between UAVs. It was concluded that linear relationship exists between speed and Consecutive TPC Ratios, and that UAV speed can be estimated using the Consecutive TPC Ratios of the CS-CLPC and FSPC algorithms.

Future work might be applying this concept to urban fading environments, applying with GPS/INS for more precise speed estimation, or finding the linearity on closed loop power control with Kalman gain step-size.

Acknowledgment

This paper was performed for the Smart UAV Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Knowledge Economy of Korea.

References

[1] J. Perez-Romero, O. Sallent, R. Agusti, and M. A. Diaz-Guerra, Radio Resource Management Strategies in UMTS. New York: John Wiley & Sons, 2005.

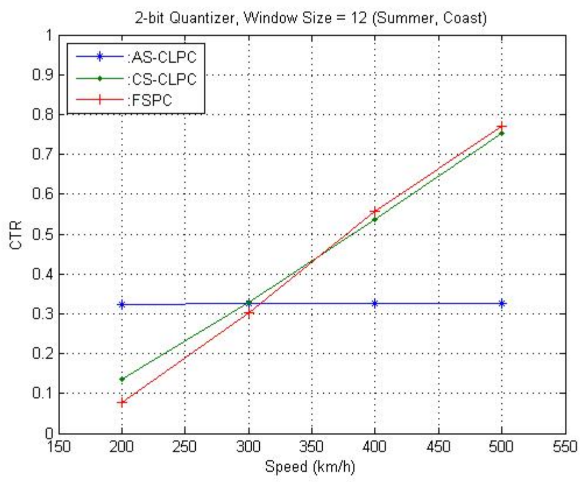


Figure 5. CTR vs. UAV1 speed at 2-bit Quantizer, window size 12 (Summer, Coast)

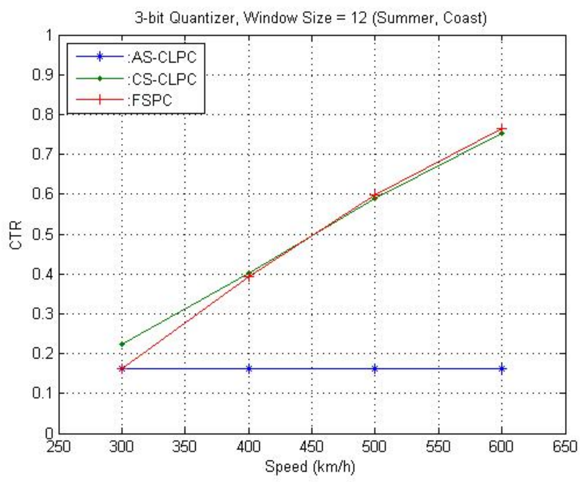


Figure 7. CTR vs. UAV1 speed at 3-bit Quantizer, window size 12 (Summer, Coast)

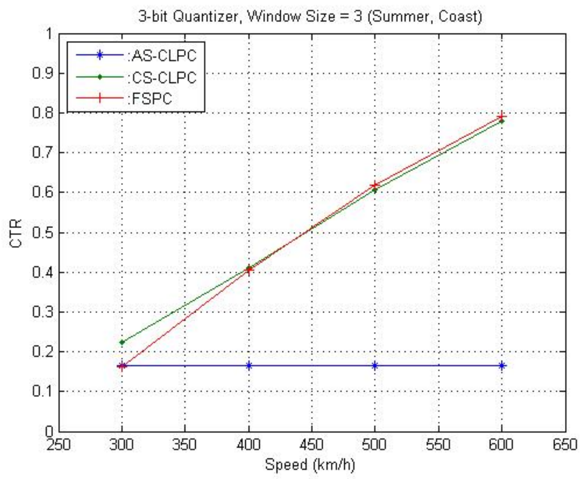


Figure 6. CTR vs. UAV1 speed at 3-bit Quantizer, window size 3 (Summer, Coast)

[2] W. Ding, J. Wang, and A. Almagbile, ”Adaptive filter design for UAV navigation with GPS/INS/Optic flow Integration,” 2000 Int. Conf. Electrical and Control Engineering (ICECE), Rome, Italy, Aug. 2010, pp. 4623-4626.

[3] R. Patachaianand and K. Sandrasegaran, ”Performance comparison of adaptive power control in UMTS,” Int. Conf. Wireless Broadband and Ultra Wideband Commun. (AusWireless 2007), Sydney, Australia, Aug. 2007, pp. 81-85.

[4] R. Patachaianand and K. Sandrasegaran, ”User speed estimation techniques for UMTS,” Electronics Letters, vol. 43, no. 19, Sep., 2007, pp. 1036-1037.

[5] R. Patachaianand and K. Sandrasegaran, ”Consecutive transmit power control ratio aided adaptive power control for UMTS,” Electronics Letters, vol. 43, no. 5, March, 2007, pp. 55-56.

[6] J. H. Kim, S. J. Lee, and Y. W. Kim, ”Performance of single-bit adaptive step-size closed-loop power control scheme in DS-CDMA system,” IEICE Trans. Commun., vol. E81-B, no. 7, July 1998, pp. 1548-1552.

[7] S. Nourizadeh, P. Taaghol, and R. Tafazolli, ”A novel closed-loop power control for UMTS,” IEE 3G 2000 Mobile Commun. Technologies (IEE 3G00), London, England, March 2000, pp. 56-59.

[8] P. Taaghol, ”Speed-adaptive power control for CDMA systems,” Bechtel Telecommunications Technical Journal, vol. 2, no. 1, Jan., 2004.

[9] H. Lee and D. H. Cho, ”A new user mobility based adaptive power control in CDMA systems,” IEICE Trans. Commun., vol. E86-B, no. 5, May 2003, pp. 1702-1705.

[10] W. Li, V. K. Dubey, and C. L. Law, ”A new generic multistep power control algorithm for the LEO satellite channel with high dynamics,” IEEE Commun. Lett., vol. 5, no. 10, Oct. 2001, pp. 399-401.

[11] D. Roddy, Satellite Communications. New Jersey: Prentice Hall, 1989.

[12] ITU-R Recommendations and Reports 2004. Sep. 2004.

A Simple Language for Describing Autonomous Agent Behaviors

Philippe Codognet
Japanese-French Laboratory for Informatics (JFLI)
CNRS /UPMC / University of Tokyo,
Tokyo, Japan
E-mail: codognet@jfli.itc.u-tokyo.ac.jp

***Abstract*— we present a simple and declarative language for describing autonomous behaviors in the paradigm of multi-agent systems. This framework is based on the notion of "goal constraints" and the satisfaction of these goals is done by iterative improvement, with the help of "repair functions" which will partially fulfill them, step by step. We can thus define both deterministic and non-deterministic behaviors. This framework is aimed at describing autonomous systems where the context is changing or unknown, thus goals can be re-evaluated or solved in different ways when changes occur. We will illustrate it with a simple example of life-like navigation behaviors for agents in unknown environments.**

Keywords*— *multi-agent systems, emergence, behavior languages, constraints, optimization.

I. INTRODUCTION

For more than a decade, multi-agent systems have become popular and widely used for various types of simulation applications [39]. Agents with autonomous behavior are now popular for simulating crowds in urban settings or emergency situations [13]. Many programming languages and frameworks have been proposed [4] from very high-level cognitive agents to low-level reactive agents. In the domain of computer graphics, where agents are used to represent autonomous characters populating a virtual world and interacting with the user, several high-level formalisms have been proposed [18,19], and other agent models have been proposed in a game-theoretic [33]. But there also exist also some lower-level frameworks such as the ABL language [20] which makes it possible to assign goals and subgoals to an agent in a procedural way and to define in such a way agent behaviors. However for autonomous agents evolving in an unknown or changing context, more abstract and declarative formulations are sometimes needed, and we will thus propose in this paper a framework for describing autonomous agent behaviors based on a declarative programming paradigm. We propose to use the formalism of Constraint Satisfaction Problems (CSP) as a general behavior description language. Constraints are used to state goals, or more exactly partial goals, that the agent has to achieve. However, the agent does not know in a deterministic way what action to apply in order to achieve a goal but maybe just have a (non-deterministic) procedure to partially fulfill it; procedure that could be applied repeatedly until the complete satisfaction of his goal or set of goals. Our approach is thus based on the idea of iterative improvement with a heuristic function, as popularized in the domain of search and optimization by the family of local search methods. It is worth noticing that the idea of using simple heuristics to guide the behavior of humans or animals has been recently proposed in many cases by both psychologists and biologists [12].

A key feature of our framework is that behaviors will not be described in a *procedural* way (i.e. stating explicitly the sequence of action needed to achieve the goal), but in a *declarative* way (i.e. stating conditions, and alternatives), by a heuristic function stating how much a goal is achieved. Then alternative configurations will be generated and the best one (i.e. the one achieving the best partial satisfaction of the goal) will be chosen for the next action, yielding thus an iterative improvement process. We believe that a declarative, nondeterministic formalism such as that of goal constraint is more powerful and easier to use than a procedural one.

As an application example, we will consider the problem of autonomous navigation of virtual creatures in a virtual world and show how algorithms derived from biologically-inspired models of navigation can be defined in our approach in order to produce life-like and robust behaviors. In the recent years, there have been a growing interest in both the animal behavior communities in considering non-trivial navigation problems, where animals or virtual agents does not know in advance the location of the goal but rather have a to explore the environment towards it, guided by a stimulus (e.g. light or smell) towards the goal (e.g. food), or just has to search some given area in an efficient way to locate an unknown goal (e.g. a prey). Our approach for defining a description language for autonomous Agent behaviors is indeed rooted in the observation that autonomous navigation can cast it as an optimization problem and we will propose a framework derived from local search techniques to efficiently obtain optimal or near-optimal trajectories. More generally, our framework can be used as a motivation architecture for virtual creatures, by considering variables for denoting internal states (e.g. energy, thirst, etc) and goal constraints for defining internal needs (e.g. the energy should stay above a certain level), routine behaviors (if the energy falls below some level, go for food), or external desired properties (e.g. stay away from predators).

The rest of the paper is organized as follows. Section 2 describes the agent model and Section 3 presents the basic ideas of constraint-based local search. Then, Section 4 defines concretely the declarative language for describing agent behaviors and Section 5 introduces the application example for autonomous navigation of agents. A short conclusion and perspectives end the paper.

II. Agent Model

Simple but interesting life-like behaviors with emergent properties, such as for instance those described in [5], should be easily implemented and tested. We will thus consider a simple model of reactive agents, who can sense their environment through a set of sensors perceiving external stimuli and who can perform actions on their environment through a set of effectors. In between some decision process will decide what behavior to perform, based on some internal state in which the agent is currently.

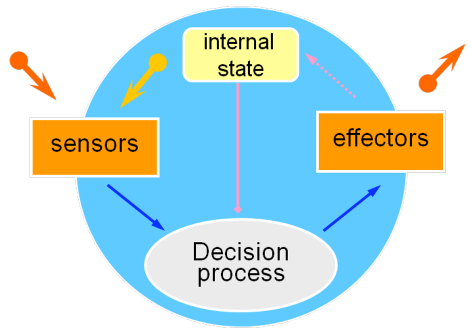


Figure 1. Simple agent model

We will however consider s slightly refined framework and detail the decision process. The agent behavior is defined by a set of *goals*, which will be specific logical relations over both the *internal variables* of the agent (defining the internal state) and the *external variables* (i.e. values of some stimulus perceived through the sensors), and thus decisions are made by a *reasoning engine*, which will try to satisfy the goals. This reasoning engine will be an iterative improvement algorithm that will be detailed in the following. Each goal will propose a repair mechanism in order to reduce the error between the current situation of the agent and a situation satisfying the goal. As an agent can have several goals to satisfy simultaneously, all the repair mechanism have to be aggregated, this will define the next action performed by the agent. This iterative step-by-step mechanism continues until the agent has satisfied all its goals. Obviously, this model can be considered as a specific instance of the Belief-Desire-Intention (BDI) agent model [25], but it is more specialized and operational.

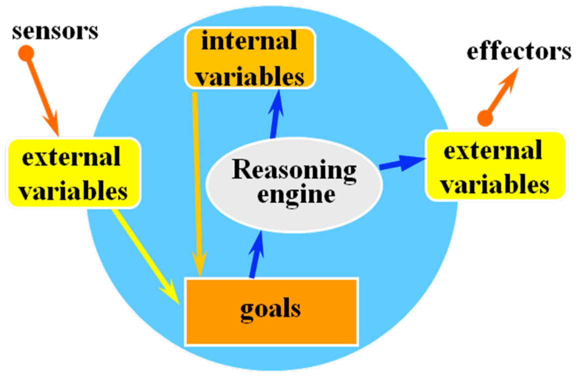


Figure 2. Refined agent model

In computer graphics and animation systems, the most common formalism for representing behaviors of high-level agents, such as virtual humans is some extension of finite state automaton (FSA) [22,40] or more complex hierarchical models [16,32]. For low-level agents, such as the swarm agents in flocks or herds and reactive agents, two basic approaches are classically used:

1. Steering behaviors, where the different low-level goals (such as grouping or escaping) are stated as forces that are then added to produce the actual behavior of the agent in a time-step manner. This approach has been pioneered by Reynolds in the late 80's [30,31], but is still active now and various extensions have been proposed [23,32,9,11, 35].
2. Particle systems [36] or potential fields [14] treating the swarm as a complex physical system.

The second approach might be interesting for efficiently simulating million of agents, but it is not flexible at all and cannot be the basis of a general behavior description framework. Indeed, elegant mathematical models of swarming behaviors have been proposed [7] but they rely on idealistic assumptions and cannot be extended to more complex models that the entomologist have nevertheless pointed out from real observations.

We are obviously closer to the first approach above, but we propose to use the formalism of *constraints* as a general behavior description language. Constraints are used to state goals, or more exactly partial goals, that the agent has to achieve. This can be seen as an extension of the steering behavior approach where constraints are solved logically instead of forces added numerically. One interesting point however is that the constraint formalism is naturally nondeterministic, as opposed to any force-based formalism such as steering behaviors, which is intrinsically deterministic. Indeed we find here again the classical dichotomy between declarative and procedural languages.

III. Goal Constraints and local search

In recent years, the interest for the family of Local Search methods and Metaheuristics for solving large combinatorial problems has been increasing and has attracted much attention from both the Operations Research and the Artificial Intelligence communities for solving real-life problems [1,10]. Local Search (i.e. non-complete) methods have been widely used in Combinatorial Optimization for finding optimal or near-optimal solutions for more than a decade [arts, gonzales], and efficient general-purpose systems for local search now exist, and Simulated Annealing, Genetic Algorithms, Tabu Search, neighboring search, Swarm Optimization, Ant-Colony optimization, etc, are all different kinds of metaheuristics that can be applied to different sets of problems ranging from resource allocation, scheduling, packing, layout design, frequency allocation, etc, in order to produce task-specific local search algorithms. These methods usually start from one random configuration (or a set of random configurations) and try to improve this configuration, little by little, through small changes in the values of the variables. Hence the term "local search" as, at each time step, only new configurations that are "neighbors" of the current configuration are explored. The definition of what constitutes a neighborhood will of course be problem-dependent, but basically it consists in changing the value of a few variables only (usually one or two, even for a problem with hundreds or thousands of variables). The advantage of Local Search methods is that they will usually quickly converge towards a solution (if the optimality criterion and the notion of neighborhood are defined correctly...) and not exhaustively explore the entire search space.

We will use the notion of *constraints* to represent at a declarative level goals that agent have to achieve. Constraints are logical relations, specific relation from a limited vocabulary, for instance artihmetic relations (e.g. =, ≤, etc) or specific symbolic relations (e.g. *all_different)* for which there exist some efficient solving algorithms. Thus achieving (logically) the goals is then reduced to solving (numerically) the constraints. We have developed in previous work [6] a framework for autonomous navigation of agents in virtual worlds based on a constraint-based combinatorial optimization algorithm named "adaptive search" and applied it to path-finding. The core ideas are:

1. To consider for each constraint a heuristic function that is able to compute an approximated degree of satisfaction of the goals (the current "error" on the constraint);
2. To aggregate constraints on each variable and project error on variables thus trying to repair first the variable with worst "error"; and
3. To update and refine these heuristic functions as the agent explores the environment and discovers new information.
4. To use some sort of "Tabu list" in order to give the agent a short-term memory and avoid having it trapped in loops and local minima.

This constraint-based local search method gave us the inspiration for the behavior description language for autonomous agents, as we would like now to rephrase this approach in a more abstract manner and generalize it to any type of behaviors, as a generic goal-based motivation architecture for autonomous agents.

IV. A Behavior Description Language

We will consider that each variable v_i in the model is either an internal variable of some agent or an external variable of the environment (also called a *stimulus*) and is ranging over a discrete or real-valued domains D_i. Each variable has thus a possibly distinct domain. A *configuration* over the domains $D_1 ,..., D_n$ is a vector $(d_1,...,d_n)$ of values with $d_i \in D_i$. When the domains are clear from the context, we will simply speak of a confiruation of size n.

An ***agent*** is defined by:

- A set of internal variables $V=\{ v_1, ..., v_k\}$ (describing the internal state of the agent) with associated domains
- A set of external variables $V'=\{ v'_1, ..., v'_j\}$ (describing the *stimuli* perceived by the agent) with associated domains
- A set of behavior goals: $G=\{ g_1, ..., g_p\}$ (defining the intended behaviors of the agent)
- An aggregation operator $\sum$ (used to combine behaviors and decide which action to perform)

The aggregation operator will be used to "sum up" the repair actions proposed to partially solve each of the p behavior goals of the agent (which could at some point be contradictory) and thereof to produce a single action to be performed by the agent. This operator is defined when programming the agent, and takes a set of p configurations to produce a single result configuration. It could be for instance a (component-wise-) sum, average, max, or any other type of function. In order to give more importance to some goals w.r.t. others, aggregation operators can be weighted sums or hierarchical (e.g. lexicographic ordering choosing to satisfy some first-ranking goal first and then the second-ranking goal only when the first one is satisfied, and so on so forth). Thus complex schemes, including for instance sequential satisfaction of goals or opportunistic behaviors, can be encoded with specific aggregation operators. As each goal g_i of the agent might involve a different set V_i of variables with $V_i \subset V$, the tuple resulting from the repair action of goal g_i has to be extended to a complete configuration of size k. This is done by completing the result vector with corresponding current values of the internal variables of the agent. Then aggregation operator $\sum$ is applied over configuration of size k only.

A ***behavior goal***, or simply a ***behavior***, is defined by:

- A set of *variables* $X \subset (V \cup V')$
- A *goal constraint* c over X
  (logical relation stating when the goal is achieved)
- A real-valued *error function* f
  (giving an heuristic value on how much the goal is unsatisfied)
- A *repair mechanism* r over X
  (to be applied when the goal is not satisfied)

Consider an n-ary behavior goal $g(X_1, \ldots, X_n)$ and associated variable domains $D_1, \ldots, D_n$. An error function f_g associated with the behavior goal g is a real-valued function from $D_1 \times \ldots \times D_n$ such that $f_g(X_1, \ldots, X_n)$ has value zero if $c_g(X_1, \ldots, X_n)$ is satisfied. Observe that it could be possible for f_g to have value zero when c_g is not satisfied, as f_g is only an approximation heuristic function representing the degree of non-satisfaction of the goal constraint. This function is intended to give an indication on how much the constraint is violated. For instance in path-planning applications and spatial goal constraints, the error function can be seen as (an approximation of) the distance between the current configuration (i.e. position) and the closest satisfiable region of the constraint domain, e.g. the air-distance. Since the error is only used to heuristically guide the agent, we can use any simple approximation when the exact distance is difficult (or even impossible) to compute.

A ***repair mechanism*** is defined by:

- A set of variables $X = \{ X_1, \ldots, X_n \}$
- A *repair generator RG*
  (generating a set of alternative configurations for variables in X)
- An *escape generator EG*
  (generating stochastic values for variables in X)

The *repair generator* will generate alternative values only for the internal variables of the agent, as external variables are considered as constants by the agent, in the hope that some of them will better satisfy the goal, that is, will produce a configuration with error less than that of the current one w.r.t. error function of the goal. Then the best one (i.e. with smallest error) will be selected as next possible action for the agent. However, it might happen no alternatives generated by the repair mechanism are improving the error with respect to the goal constraint. Then the *escape generator* will be used in order to propose a stochastic action. In that case the escape generator is used to produce a random value for the internal variables, which will be used for the next action. The range in which this stochastic choice should be applied is of course dependent on the behavior, this is why the escape generator is part of the repair mechanism of each given behavior. Observe that a simple and conservative (although a bit stupid…) escape strategy is to have the identity as escape generator, i.e. to keep the current values of the variables. Therefore, the agent will freeze and do nothing, hoping that the environment will change and allow for some action at a later time. On the other hand a more complex escape generator (e.g., a stochastic procedure based on Perlin noise or Levy flight) will provide a non-deterministic behavior.

As a very simple example, the behavior of an agent which should go to a given position (already known) can be modeled by a goal constraint on a single internal value, its position:

$$agent.position = target.position$$

The error function could be the air-distance:

$$|agent.position - target.position|$$

The repair mechanism can generate several alternative positions in front of the agent for the next step, as there might be some obstacles and the direct way (straight line) might be infeasible. In any case the agent will select the best remaining position (closest to the target).

Observe that if all the repair generators are deterministic (i.e. generate a single alternative configuration), then the overall behavior of the agent will be deterministic. For example the framework of Steering behaviors [30,31], is based on the notion of steering forces that are added to produce the overall behavior of the agent. In the case of so-called *boids* with flocking behaviors steering forces are three distinct separation, alignment and cohesion forces that are computed and then added component-wise, thus deterministically. Indeed, comparing to the steering behavior paradigm, we propose here non-deterministic behaviors and we do not try to define a combination of forces that will bring the agent to the desired position but just check possible alternative configurations and choose the best one, which can be done efficiently in the local search approach. Also observe that we can cope with dynamically changing priorities between behaviors (e.g. in the definition of the aggregation operator of the agent), because the error functions are re-evaluated and combined at each time step.

The overall generic computation model for the agents can thus be simply defined as follows:

```
For each (agent A in the world) {
  For each (goal G of agent A){
    Evaluate current error $F_G$
    Evaluate satisfaction condition $C_G$
    If (not $C_G$) {
      Generate a set of alternatives R
      For each (alternative S in R){
        Evaluate error function $F_G$ on S}
      Select best alternative config. S′
             (i.e. with smallest error)
      If (error of S′ > $F_G$) {
        S′ = escape generator for G}
    repair action $R_G$ = S′ extended to $V_A$
    }
  }
  Aggregate all repair actions with ∑
  Perform resulting action for agent A
```

}

It is clear that the combination of several goals might produce quite complex behaviors. For instance if the agent should go towards some object (goal constraint: *agent.position = target.position*), and stay at some distance of it (goal constraint: *agent.position ≥ target.position + d*), then a following behavior (at distance *d*) will be simply obtained if the target is moving.

V. EXAMPLE: NAVIGATION STRATEGIES

For the last two decades, the observation and modeling of animal motion and navigation strategies by zoologists and entomologists have inspired researchers in Artificial Intelligence and Artificial Life for the design of simulation models for crowds and autonomous agents. In particular, Nature-inspired simulations based on the metaphor of *swarm intelligence*, such as the foraging simulation of ant-colonies [3] or collective motion such as flocks schools and herds [30,8]. More recently, [37] presents a comprehensive review the similarities between collective behaviors of different types of agents: bacterial colonies, cells, insects, fishes, birds, mammals, humans, etc. Besides these collective motion and cognition, researchers have also been investigating movements and navigation strategies of single animals, especially the navigation patterns of predators looking for a prey in an unknown environment. It appears that similar patterns and probably search strategies are put at use by very different animal (e.g. bees, flies, moths, peacocks, albatross, and can be modeled by so-called “Levy flights” [29]. This method has been defined as an optimal search behavior (e.g. better than random walk or so-called spiral search) for random search in an unknown environment, especially when the targets are sparse [38] [2]. Even more interestingly [27] showed that Levy flight navigation strategies could be an emergent property resulting from a simple gradient-following strategy for chemotaxis (reaction to some chemical stimulus). This is indeed the behavior of simple living creatures such as the small soil nematode *Caenorhabditis elegans* [21,24]. Very recently [15,28] showed that a simple gradient-following strategy such as chemotaxis can be used to navigate in complex mazes and showed that such simple chemotaxis finds in several examples the shortest route, although this method does not guarantee to always find an optimal path.

When considering the domain of adaptive behaviors [34], an exploration process guided by a stimulus (e.g. light or smell) towards a goal (e.g. food) of unknown location, two different methods are usually defined: temporal difference or spatial difference. The temporal difference method consists in considering a single sensor (e.g. the nose) and checking at every time-point the intensity of the stimulus. If the stimulus is increasing, then the agent continues in the same direction, otherwise the direction is changed randomly and so on so forth. This is exemplified by the chemotaxis of the *Caenorabditis elegans*. With this method, the agent eventually reaches the goal, but might wander in some irrelevant regions of the environment in between [21]. A more efficient strategy is possible by using the spatial differences method [17]. It requires to have (at least) two identical sensing organs, placed at slightly different positions on the agent (e.g. the two ears). The basic idea is to favor, at any time-point, motion in the direction of the sensor which receives the most important stimulus. If none of the sensors perceives an increasing of the stimulus, then a random move is performed. This behavior gives very good results and the agents goes most of the time directly towards the goal. Also when the goal is moved away, e.g. because the prey is moving, the agent reacts instantly (as the stimulus is checked iteratively at every time-point) and moves towards the new location.

Both of these models fit quite well within the general framework for describing agent behaviors presented in the previous section. The temporal difference method consists in having a single internal variable for storing the value of the stimulus at the previous time point and if the current value of the stimulus is not increasing then a random turn (within some given limits, e.g. plus or minus 30 degrees) will be performed. The spatial difference method consists in having two internal variables for checking the values of the stimulus with two sensors in different positions and then making a move in the direction of the sensor with highest stimulus. Observe that one can have more than two sensors and thus check the intensity of the stimulus at various alternative points, but it seems than it does not improve the search process significantly [17], explaining thus maybe why animals and humans only need two symmetric sensors (e.g. eyes or ears).

VI. CONCLUSION

We presented a simple and declarative framework for describing autonomous behaviors in the paradigm of multi-agent systems. This model is a generalization of previous work on navigation models for autonomous agents in virtual worlds [6], and can be used as generic goal-based motivation architecture for autonomous agents. Our language is based on the notion of “goal constraints” and the satisfaction of these goals is done by iterative improvement, with the help of “repair functions” which will partially fulfill them, step by step. We are currently finishing an implementation of this framework using the *processing* language [26].

REFERENCES

[1] E. Aarts and J. K. Lenstra, Eds., *Local Search in Combinatorial Optimization,* Chichester, UK: John Wiley and Sons, 1997.

[2] F. Bartumeus and J. Catalan, “Optimal search behavior and classic foraging theory”, *Journal of Physics A*, vol. 42 (2009), 434002.

[3] E. Bonabeau, M. Dorigo and G. Theraulaz*, Swarm Intelligence: From Natural to Artificial Systems,* Oxford University Press 1999.

[4] R. H. Bordini, M. Dastani, Mehdi; A. El Fallah Seghrouchni (Eds.), *Multi-Agent Programming: Languages, Platforms and Applications*, Springer Verlag 2005.

[5] V. Braitenberg, *Vehicles - Experiments in Synthetic Psychology*, MIT Press, 1984.

[6] P. Codognet, "Animating virtual creatures by constraint-based adaptive search" In: *Proc. VSMM2000, 4th Int. Conf. on Virtual Systems and Multimedia*, Gifu, Japan, IOS Press 2000.

[7] F. Cucker and S. Smale, "Emergent Behavior in Flocks", *IEEE Transactions on Automatic Control*, vol. 52, Issue 5 (2007), pp 852 – 862.

[8] I. Couzin, J. Krause, R. James, G. D. Ruxton, and N.R. Franks, "Collective Memory and Spatial Sorting in Animal Groups", *Journal of Theoretical Biology*, vol. 218, no. 1, Pages 1-11.

[9] I. D. Couzin, "Collective cognition in animal groups", *Trends in Cognitive Sciences*, vol. 13 no. 1, 2008, pp 36-43.

[10] T. Gonzalez (Ed.), *Handbook of Approximation Algorithms and Metaheuristics*, Chapman and Hall / CRC, 2007.

[11] C. Hartman and B. Benes, "Autonomous Boids", *Computer Animation and Virtual Worlds*, vol. 17 no. 3-4 (2006), pp 199–206.

[12] Hutchinson, J.M.C., Gigerenzer, G., 2005. "Simple heuristics and rules of thumb, where psychologists and behavioural biologists might meet", *Behavioral Processes*, vol. 69, no. 2, 2005, pp 97–124.

[13] T. Ishida, L. Glasser and H. Nakashima, *Massively Multi-Agent Systems*, LNAI 3446, Springer Verlag 2004.

[14] X. Jin, C. C. L. Wang, S. Huang and J. Xu, "Interactive control of real-time crowd navigation in virtual environment", In: *Proc. 2007 ACM symposium on Virtual Reality Software and Technology*, pp 109-112, ACM Press 2007.

[15] I. Lagzi, S. Soh, P. J. Wesson, K. P. Browne and B. A. Grzybowski, "Maze Solving by chemotactic droplets", *Journal of the American Chemical Society*, vol. 132 no. 4, 2010, pp 1198–1199.

[16] F. Lamarche and S. Donikian, "Crowd of Virtual Humans: a New Approach for Real Time Navigation in Complex and Structured Environments", *Computer Graphics Forum* vol. 23 no. 3, 2004, pp. 509-518.

[17] W. Leow. "Computational Studies of Exploration by Smell", in [34].

[18] P-S. Liew, C-L. Chin and Z. Huang, "Development of a computational cognitive architecture for intelligent virtual character", *Computer Animation and Virtual Worlds*, vol. 20 no. 2-3 (2009), pp 257-266.

[19] L. Luo, S. Zhou, W. Cai, et al., "Agent-based human behavior modeling for crowd simulation", *Computer Animation and Virtual Worlds*, vol. 19 no. 3-4 (2008), pp 271-281.

[20] Michael Mateas and Andrew Stern, "A Behavior Language: Joint Action and Behavioral Idioms", In: *Life-like Characters: Tools, Affective Functions and Applications*, H. Prendinger and M. Ishizuka (Eds.), Springer Verlag 2004.

[21] T. Morse, T. Ferrée, S. Lockery, "Robust Spatial Navigation in a Robot Inspired by Chemotaxis in C. Elegans", in [34].

[22] T. Noma, L. Zhao and N. I. Badler, "Design of a Virtual Human Presenter", *IEEE Computer Graphics and Applications*, vol. 20, no. 4, 2000, pp. 79-85.

[23] C. Pedica and H. Vilhjalmsson , Social Perception and Steering for Online Avatars, *In: Proc. IVA'08, 8th international conference on Intelligent Virtual Agents*, Tokyo, Japan, LNAI 5208, Springer Verlag 2008, pp 104 – 116.

[24] J. T. Pierce-Shimomura, T. M. Morse, and S. R. Lockery, "The fundamental role of pirouettes in Caenorhabditis elegans chemotaxis", *Journal of Neurosciences*, vol. 19 no. 21, pp 9557–9569.

[25] A. S. Rao and M. P. Georgeff, "Modeling rational Agents with a BDI-Architecture", In : *Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 1991, pp 473-484.

[26] C. Reas and B. Fry, *Processing*, MIT Press 2007.

[27] A. M. Reynolds, "Deterministic walks with inverse-square power-law are an emergent property of predators that use chemotaxis to locate randomly distributed prey", *Physical Review E*, vol. 78 (2008), 011906

[28] A. M. Reynolds, "Maze-solving by chemotaxis", *Physical Review E*, vol. 81 (2010), 062901.

[29] A. M. Reynolds and C. J. Rhodes, "The Levy flight paradigm: random search patterns and mechanisms", *Ecology*, vol. 90 no. 4, 2009, pp 877-887.

[30] C. W. Reynolds, "Flocks, Herds, and Schools, A Distributed Behavioral Model", *Computer Graphics,* vol. 21 no. 4 (SIGGRAPH '87 Conference Proceedings), 1987, pp. 25-34.

[31] C. W. Reynolds, "Steering behaviors for autonomous characters", In: *Proc. of 1999 Game Developers Conference*, San Francisco, Miller Freeman Group 1999, pp. 763-782.

[32] W. Shao and D. Terzopoulos, "Autonomous pedestrians", *In: Proc. of the ACM SIGGRAPH'2005, Symposium on Computer Animation*, ACM Publishing 2005.

[33] S. Schiffel and M. Thielscher, "A Multiagent Semantics for the Game Description Language", In: *Agents and Artificial Intelligence,* Computer and Information Science series vol. 67*,* Springer Verlag, 2010, pp 44-55.

[34] N. Schmajuck (Ed.), "Special issue on Biologically-inspired models of Navigation", *Adaptive Behavior*, vol. 6, no. 3/4, Winter/Spring 1998.

[35] M. Schuerman, S. Singh, M. Kapadia and P. Faloutsos, "Situation agents: agent-based externalized steering logic", *Computer Animation and Virtual Worlds*, vol. 21 no. 3-4 (2010), pp 267–276.

[36] A. Treuille, S. Cooper and Z. Popovic, "Continuum crowds", *ACM Transactions on Computer Graphics*, vol. 25 no. 3 (SIGGRAPH 2006 conference proceedings*)*, 2006, pp. 1160-1168.

[37] T. Vicsek and A. Zafiris, "Collective Motion", preprint available on *arXiv:1010.5017v1 [cond-mat.stat-mech]*.

[38] G. M. Viswanathan, S. V. Buldyre1, S. Havlin, M. G. E. da Luz, E. P. Rapos and H. E. Stanley, "Optimizing the success of random searches", *Nature*, no. 401 (28 October 1999), pp 911-914.

[39] Michael Wooldridge, *An Introduction to MultiAgent Systems*, John Wiley & Sons Ltd, 2002.

[40] H. Xiao and C. He, "Behavioral Animation Model for Real-Time Crowd Simulation", In: *Proc. of 2009 International Conference on Advanced Computer Control*, IEEE Press 2009.

Mobile Robots Path Planning using Genetic Algorithms

Nouara Achour
LRPE Laboratory, Department of Automation
University of USTHB
Algiers, Algeria
nachour@usthb.dz

Mohamed Chaalal
LRPE Laboratory, Department of Automation
University of USTHB
Algiers, Algeria
mchaalal@usthb.dz

***Abstract*—In this article, we discuss path optimization to solve the problem of path planning for autonomous mobile robots. We consider the case of constrained environments where the robot is represented as a point. For that, we used an approach based on models of evolution; the genetic algorithms which are an interesting alternative to conventional methods of path planning. A population of paths is obtained firstly using a random distribution strategy. The performance of the proposed Genetic Algorithm based approach is tested on environments with increasing complexity. Through some results, we give a comparison between this strategy and a method based on Lazy A* search.**

Keywords-path planning; PRM; Genetic algorithms; robotics.

I. INTRODUCTION

The aim of motion planning is to find the allowable movements of a robot in a constrained environment (clearance of a robot from obstacles is low). In its simple version, it consists of finding a path free of collision from an initial configuration q_{init} to a final configuration q_{final}, it is a PSPACE-hard difficulty as shown by Reif [1].

This means that the complexity of the path planning problem increases exponentially with the dimension of the configuration space. Based on scientific research over the past twenty years, we found that there are two major families of algorithms that address this problem. One uses a deterministic approach while the second uses a probabilistic approach [2]. There are several deterministic search algorithms; we can cite Bellman, breadth first search, etc.

Recently, random sampling has emerged as a powerful technique for planning in large configuration spaces [3][4]. Random-sampling planners are classified into two categories: PRM (Probabilistic RoadMap) and RRT (Rapidly-Exploring Random Tree).

An other approach uses genetic algorithms which are Meta-heuristic search algorithms. Genetic algorithms (GA's) are search strategies based on models of evolution [5]. They have been shown to be able to solve hard problems in tractable time. Here, we need a solution space composed of a set of nodes randomly generated in C_{free} (free Configuration space).

When using GA's, we need a solution space composed of a set of nodes randomly generated in C_{free}. The algorithm execution is performed by the research of the best configurations between the initial and the final nodes with checking the optimization criterion which is here the distance.

The strength of this method is that it allows to explore and exploit the best solutions by two operators, which are selection and genetic reproduction.

Section II gives an overview of previous related works and exposes this work's motivations, Section III lays the mathematical description of the terms and concepts used in this article, Section IV describes briefly the PRM-based path planning, Section V details our approach in using genetic algorithms to plan optimized paths. In the last section, we report a series of actual runs.

II. RELATED WORK

GA's are considered as optimization algorithms for search in a space of potential solutions, so they are faced with the exploration-exploitation dilemma. The solution to the problem of planning by Genetic Algorithms is proposed for the first time by [5]. There are also other contributions by several researchers [6][7]. The common problem to all methods is how to choose the initial population. Most of these methods use a set of paths encoded in the chromosomes. The optimal path is calculated after several iterations. The necessary step in these algorithms is the determination of the fitness function (optimization criterion). [4] proposed to use a function of performance determined by the linear combination of distance, the smoothing angle and the robot position from the obstacles. Some papers have focused on dynamic environments [8] and others have tried to investigate in the synergism of respectively fuzzy logic - GA's [10] and neural networks-GA's [11].

In this article, we discuss path optimization to solve the problem of path planning for autonomous mobile robots. We have mainly focused on using genetic algorithms to calculate optimized paths, we have thus demonstrated their advantages and disadvantages, and for this we made a comparison with a widely used approach, PRM associated with A^* algorithm for finding optimal paths.

III. PRELIMINARIES

As the paper covers several notions relating to C-space (configuration space), we give some definitions for the most pertinent ones used here.

Definition 1: A **Workspace** $\mathcal{W}$ is a physical space represented by $\mathbb{R}^2$ for planar (2D) or $\mathbb{R}^3$ for 3D spaces.

Definition 2: An ***Obstacl***e $\mathcal{O}$ is a portion of $\mathcal{W}$ that is "permanently" occupied, represented by the *obstacle region* $\mathcal{O}, \mathcal{O} \subseteq \mathcal{W}$.

Definition 3: A robot $\mathcal{A}$ consists of one rigid body, or more "motion-constrained" rigid bodies, represented by the *robot region* $\mathcal{A}$ and is the set of all points in $\mathcal{W}$ that lie in $\mathcal{A}$, $\mathcal{A} \subseteq \mathcal{W}$.

Definition 4: A robot configuration q is a set of parameters that completely specify the position of robot $\mathcal{A}$ with respect to a fixed frame F, $\mathcal{A}(q)$ is the region of $\mathcal{W}$ occupied by $\mathcal{A}$ at configuration q. Each parameter of q corresponds to one degree of freedom of the robot.

Definition 5: The configuration space $\mathcal{C}$ for a robot $\mathcal{A}$ is the set of all configurations of robot $\mathcal{A}$ in $\mathcal{W}$ or:

$$\mathcal{C} = \{\forall\ q\ |\mathcal{A}(q) \subseteq \mathcal{W}\} \qquad (1)$$

Definition 6: The configuration space obstacles $\mathcal{CO}$ is the mapping of the obstacles in the workspace to the configuration space, it is the set of all configuration of robot $\mathcal{A}$ at which the robot region is in contact or overlaps with obstacles regions or:

$$\mathcal{CO} = \{all\ q, i\ |\mathcal{A}(q) \cap \mathcal{O}_i \neq \emptyset\} \qquad (2)$$

Definition 7: Free configuration space $\mathcal{C}_{free}$ is the set of configurations at which the robot is free from collision with the Workspace obstacles, or simply:

$$\mathcal{C}_{free} = \mathcal{C} \backslash \mathcal{CO} \qquad (3)$$

Definition 8: A local-path lp is a continuous function of a parameter s which takes values in the interval : [0, 1] to $\mathcal{C}$ or: $lp: [0, 1] \rightarrow \mathcal{C} \mid lp(s)=q(s)$.
Furthermore, if the local-path is defined by its end configuration $lp_{(q0,q1)}$, then $lp(0)= q_0$, $lp(1)= q_1$.

Definition 9: The general definition of a path p is the same as that of a local-path, and when it is additionally defined by its start and end configurations $p_{(qs,\ qe)}$, then $p(0)= q_s$, $lp(1)= q_e$.

Given the discrete search used here, p will be mostly defined as an ordered set of k local-paths, or $p_{(lp1,\ lp2,..\ lpk)}$ where; $lp_i(1)= lp_{i+1}(0)$, $i=1,..k-1$, this results in a continuous path. A $sub\text{-}path_{(q1,q2)}$ of a path p is a continuous function from $[s_1, s_2] \rightarrow \mathcal{C}$ where $p(s_1) = q_1, p(s_2) = q_2$ and $\forall s \in [s_1, s_2]$: $sub\text{-}path_{(q1,q2)}\ (s) = p(s)$. $0 \leq s_1 < s_2 \leq 1$.

IV. PLANNING WITH PRM

Probabilistic algorithms are based on the use of randomness for the construction of a graph capturing, in condensed form, the connectivity of the free space C_{free}, thereby precluding any explicit representation of the configuration space C. The basic idea of planners based on random sampling is to exploit the outstanding performance of collision detection algorithms that verify whether a given configuration is free or not. It should be noted that the completeness of these algorithms is rather low.

The principle used to sample the space C_{free} is that of uniform random sampling, which can blanket the entire free region.

```
Simple Query PRM Algorithm (N, B,M,$\mathcal{V}$'s, $q_{ini}$ , $q_{fin}$)
1. Sample N configurations
2. R(V,E)←For every node n, assign the closest B visible
   nodes as visible from n.
3. Add $q_{ini}$ and $q_{fin}$ to R and query it using Lazy A* for
   path p, if successful return p.
4. Sample and connect M new nodes to R in "difficult"
   regions.
5. Query R using Lazy A* for path p, if successful return
   p else return Failure.
```

We may note that in step 1 and 2 we only construct a roadmap without actually checking the nodes in V nor the edges in E for collision. This will be performed during the query phase only when necessary by using a Lazy A*[9] with weighed L-infinity norm $d(q', q'')_{wL\infty}$ (the weights being ν_i's) as the cost of edge $e(q',q'')$. the difficult regions in step 4 are those collision free nodes that fail more often to connect to their neighbouring nodes compared to others, this could potentially indicate that these nodes are located in narrow regions [9][12] (Figure 1).

V. PATH PLANNING USING GENETIC ALGORITHMS

The algorithm is divided into two phases; first, we generate a set of configurations in free space and then in the second phase we use genetic algorithms as a metaheuristic search procedure to find the optimal path that leads robot from q_{ini} and q_{fin}. In order to sample the space C_{free} , we use the uniform random sampling, which can blanket the entire free region. Each time we generate a configuration, we check whether it is in collision or not. In our case, we generate an initial population of segments with a performance function based on the minimum distance (Figure 2).

To find the optimal path by the approach of genetic algorithms, we try to find the best segments of this path, such that the sum of their metric distance is minimal. The coding of these solutions is as follows: each chromosome contains two nodes, an initial and final node. These nodes are those generated in the first phase, so they are free of collision.

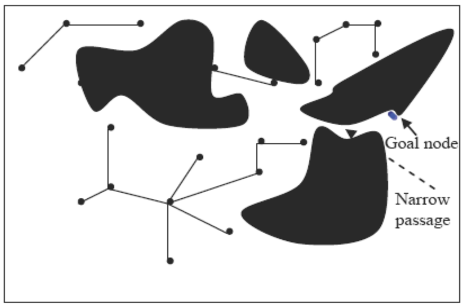


Figure 1. Roadmap built by PRM in a C-space with narrow regions.

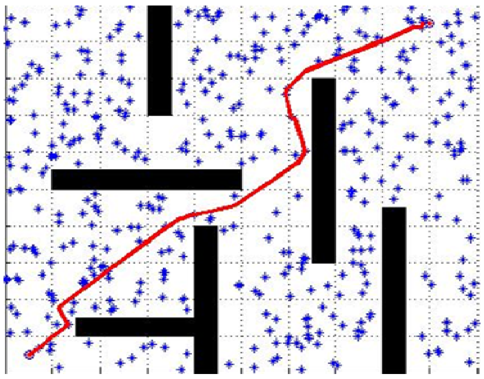


Figure 2. The blue stars represent all the configurations (nodes) generated randomly and the best path (in red) passing through the best nodes.

The algorithm runs in a finite number of iterations. At each iteration, we choose a new population that depends on the initial node of each segment (which represents the final node of the best segment in the previous iteration). The algorithm stops when we reach the goal configuration.

In order to be eligible in the initial population, a segment should not collide with obstacles.

Solutions encoding is performed by real numbers to store the topological structure of the solution space to the genotype space. With binary coding, there is a risk of losing information and the concept of geometric path.

After the initialization of the solution space, the second operation *crossover* is performed. This operation can produce new solutions from the initial population, which increases the chance of finding the best segment. The mathematical formulation of the operation of crossing depends on the nature of representation used to encode the solutions, for example, the crossing at one point can be applied to individuals with binary representation or real numbers. It consists of choosing a random point for the two chromosomes to exchange genetic material between two people around this point.

The crossing is performed only for the final node of each chromosome (Figure 3), and each time we check whether the node is in collision or not and visible to the initial node to avoid collision with obstacles.

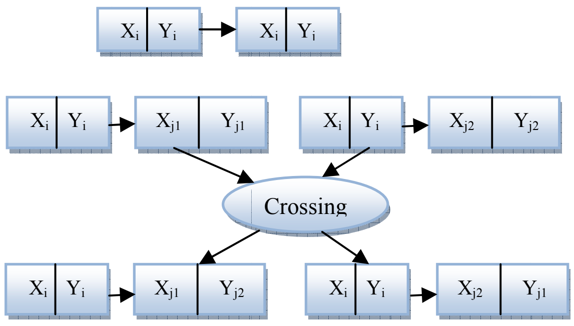


Figure 3. Crossing is performed only for the final node of each chromosome.

The probability of crossing and mutation are respectively equal to 1 and 0.

The fitness function is as follows:

$$f(s) = \left(\frac{1}{d_i} e^{\frac{-\theta_i}{d_i}}\right) - e^{\frac{-d_i}{\theta_i}} \tag{4}$$

$$d_i = \|n_i - n_{best}\|^2 + \|n_{fin} - n_i\|^2 \tag{5}$$

$$\theta_i = \tan^{-1}\frac{y_{n_{fin}} - y_{n_i}}{x_{n_{fin}} - x_{n_i}} \tag{6}$$

n_i and n_{best} being respectively the final and initial node of each segment, n_{fin} is the final node of the global path.

After evaluating each chromosome by the performance function, the best segment is the one which minimizes the distance of the path passing through these intermediate nodes. The best chromosome is determined after each generation, the final node of this chromosome becomes the starting point of the next segment.

The best node is the one which has the maximal fitness function. In our case, we have not considered the mutation operation since its random effect may produce undesirable results, there is a risk of losing the best individuals.

```
Genetic Algorithm (N, B,M,$f$, $q_{ini}$ , $q_{fin}$)
1.  Encoding solutions into vectors $(x_i, y_i, x_j, y_j)$
2.  Crossing (selection of crossover point randomly)
3.  Evaluate all chromosomes by the fitness function $f$
4.  Select the best segment whose $f$ is minimal $(x_i, y_i, x_{best}, y_{best})$
5.  Add $(x_{best}, y_{best})$ to p
6.  $x_i = x_{best}$, $y_i = y_{best}$ until $x_i = x_{fin}$, $y_i = y_{fin}$
```

VI. RESULTS

The two approaches of path planning PRM/Lazy A* and Genetic Algorithms were implemented in Matlab and have been tested on environments with simple structure then with more complexity.

In our case the mobile robot $\mathcal{A}$ navigates in a 2D environment consisting of walls and polygonal obstacles, it has three dof (two degrees for translation and one degree for rotation).

For both approaches, we considered the same initial and final configurations, namely: $q_{init} = [x_{init}, y_{init}, \theta_{init}]^t = [3, 2, \pi]^t$, and $q_{fin} = [x_{fin}, y_{fin}, \theta_{fin}]^t = [18, 19, \pi]^t$. The number of samples (M) and neighbors (B) are set respectively at 500 and 5.

Figures 4 (a, c, e, g) and (b, d, f, h) show the results obtained respectively for the first approach (PRM/Lazy A*) and second approach (GA's).

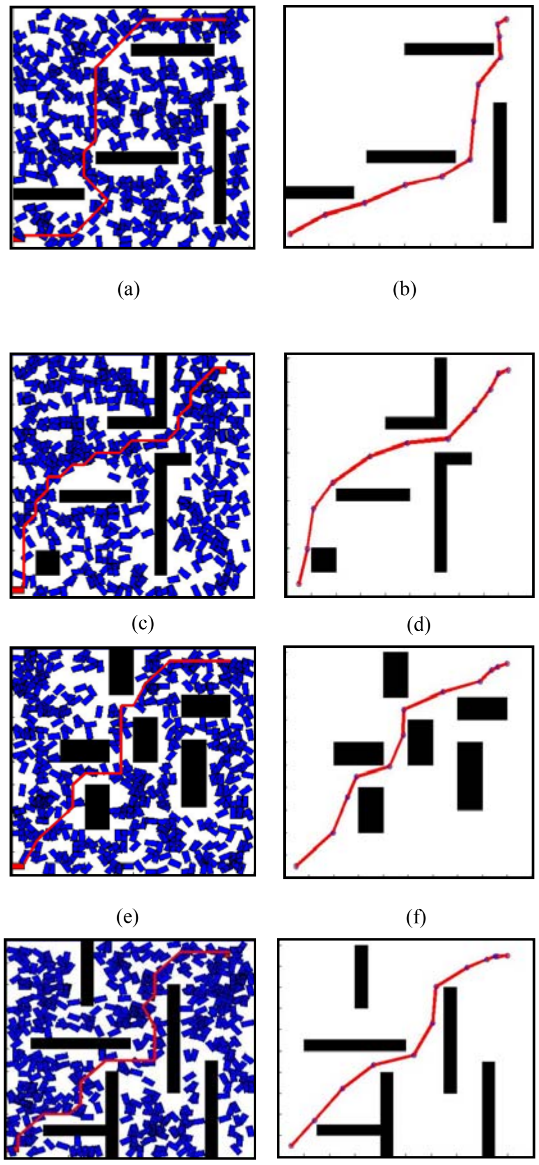


Figure 4. From top to bottom, the two paths obtained (on the right PRM/Lazy A* and GA's on the left), for environments 1, 2, 3 and 4.

Table I gives the computation time of a path by the two approaches in the four environments.

TABLE I.

	PRM/Lazy A*			GA's		
	M	N	T	M	N	T
Envt 1	500	500	118.90	500	34	20.02
Envt 2	500	1015	122.63	500	38	23.85
Envt 3	500	1634	168.92	500	41	21.96
Envt 4	500	2043	168.36	500	70	24.02

VII. DISCUSSION

In the typical results shown in Figure 2, we find that the final path is always optimized (in length) when using genetic algorithms. The time required to find the optimal path is about 24 seconds with a number of samples equal to 500.

If we increase the complexity of the environment, we obtain a path generally better optimized in distance and also in computation time. The number of iterations (70) is significantly lower than the one obtained from PRM (2043). Even if we increase the number of samples, the optimal path keeps the same performance.

The GA's approach does not require the construction of the graph of visibility, it minimizes the computation time of the path, it is very interesting for solving NP-hard problems.

Despite the good results, there remains a problem when meeting environments with narrow regions (Figure 4(c, d)) which makes very difficult the search for configurations that optimize the way, the algorithm falls into a local minimum. In this case, a probable solution is to generate additional configurations in these regions. But this is not always the right solution, because if we increase the number of configurations, there is a risk of not finding the way, despite the existence of feasible configurations in C_{free}. The failure to find a solution path is often due to the inefficiency of collision detection. By making several attempts, we can reach an acceptable solution. For example, for the environment 4, the path is found after 5 tests and 70 iterations.

VIII. CONCLUSION AND FUTURE WORKS

This paper investigates the application of GA's for solving the problem of path planning for mobile robots. We studied two approaches based on randomized algorithms, PRM and GA's. For the first approach, the results showed that we can find feasible paths for several types of environments, however, this does not qualify as 'robust' this approach, since we encounter cases where we can not find solutions, despite that there are configurations eligible to generate a path. In the second approach, which is based on genetic algorithms, a population of paths is obtained firstly using a random distribution strategy. The performance of the proposed Genetic Algorithm based approach is tested on environments with increasing complexity. The results obtained by this approach show the effectiveness of GA's, these algorithms can find the optimal path in a very short time and has the capacity to enrich the configuration space by a different set of eligible movements by using the crossover operator and selection.

As future works, we would like to extend this approach to multiple cooperating robots and mobile manipulators.

IX. REFERENCES

[1] J.H. Reif "Complexity of the mover's problem and generalizations," Proc. IEEE Transactions on Robotics and Automation, pp. 421–427, 1979.

[2] J. C. Latombe, Robot Motion Planning, Norwell, MA: Kluwer, 1991.

[3] E. Kavraki, P,Svestka, J-C. Latombe, and M.H. Overmars "Probabilistic Roadmap for path planning in high-dimensional

configuration spaces". IEEE Trans, Robot & Autom, 12(4),pp. 556–580, June 1996.

[4] L. Kavraki, M. Kolountzakis, and J.-C. Latombe. "Analysis of probabilistic roadmaps for path planning". In IEEE Trans. Robot. & Autom., volume 14, pp. 166–171, 1998.

[5] J.H. Holland, Adaptation in natural and artificial systems, Ann Arbor: University of Michigan Press, 1975.

[6] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. 1st ed. MA: Addison-Wesley, 1989.

[7] C. E. Thomas, M. A. C. Pacheco, M.M., and B.R.Vellasco, "Mobile Robot Path Planning Using Genetic Algorithms,". In foundations and tools for neural modeling, vol. 1606/1999, Springer Berlin/ Heidelberg, pp. 671–679, 1999.

[8] S. Yang , H. Cheng, and Fang Wang, "Genetic Algorithms With Immigrants and Memory Schemes for Dynamic Shortest Path Routing," IEEE Trans on Syst man, and cyb, vol. 40, n°1, pp. 52–63, 2010.

[9] R. Guernane, and N. Achour, "An Algorithm for Generating Safe and Execution-Optimized Paths" in Proceedings of the Int. Conference on Autonomic and Autonomous Systems, pp. 16–21, Spain, 2009.

[10] C. Fayad and P. Webb, "Development of a hybrid crisp-fuzzy logic algorithm optimised by genetic algorithms for path-planning of an autonomous mobile robot" in Journal of Intelligent & Fuzzy Systems, pp. 15-26, 2006.

[11] N. Noguchi and H. Terao "Path planning of an agricultural mobile robot by neural network and genetic algorithm" in Journal Computers and electronics in agriculture, 1997, vol. 18, n^{o}2-3, pp. 187–204.

[12] R. Bohlin, and L. E Kavraki, "Path Planning Using Lazy PRM" in Proceedings of the IEEE Int. Conference on Robot & Autom, pp. 521–528, IEEE Press, San Fransisco, 2000.

Utility Function-based Workload Management for DBMSs

Mingyi Zhang[†], Baoning Niu[§], Patrick Martin[†], Wendy Powley[†]
[†]Queen's University, Kingston, ON, Canada
[§]Taiyuan University of Technology, China
{myzhang, niu, martin, wendy}@cs.queensu.ca

Paul Bird, Keith McDonald
Toronto Software Lab, IBM Canada Ltd.
Markham, ON, Canada
{pbird, kmcdonal}@ca.ibm.com

***Abstract*—In the practice of autonomic computing, utility functions have been applied for achieving self-optimization in autonomic computing systems. Utility functions are suggested to guide an autonomic computing system to optimize its own behavior in accordance with high-level objectives specified by the system administrators. In this paper, we present two concrete examples to illustrate how utility functions can be properly defined in database workload management systems. Our utility functions help the systems translate high-level workload business importance policies into low-level policies of database system tuning actions, and therefore ensure the workloads achieve their required performance objectives. We also discuss the fundamental properties of a proper utility function that can be used for building autonomic workload management for database management systems.**

Keywords-Workload Management; Database Management Systems; Utility Functions; Autonomic Computing

I. INTRODUCTION

A database workload is a set of requests that have some common characteristics such as application, source of request, type of query, priority, and performance objectives (*e.g.*, response time goals) [3]. Workload management in database management systems (DBMSs) is a performance management process whose primary objective is to minimize the response time of critical queries, such as the ones for directly generating revenue for business organizations, or those issued by a CEO or VP of the organizations, maintain DBMSs running in an optimal state (*i.e.*, neither under-utilized nor overloaded), and maximize throughput of the system under these constraints. For both strategic and financial reasons, business organizations are consolidating individual database servers onto a shared data server to serve as the single source of corporate data. As a result, multiple types of requests are mixed and present on a single data server simultaneously. Request types include on-line transaction processing (OLTP) requests that are typically short and efficient, consume minimal system resources, and complete in sub-seconds, as well as business intelligence (BI) queries that can be longer, more complex and resource-intensive, and may require hours to complete. Requests generated by different applications or initiated from different business units may have unique performance objectives that are normally expressed in terms of service level agreements that must be strictly satisfied for business success.

Multiple requests running on a data server inevitably compete for shared system resources, such as system CPU cycles, buffer pools in main memory, disk I/O, and various queues in the database system. If some requests, for example, long BI queries, are allowed to consume a large amount of system resources without control, the concurrently running requests may have to wait for the queries to complete and release their used resources, thereby resulting in the waiting requests missing their performance objectives and the entire data server suffering degradation in performance. Moreover, requests presenting on a data server can fluctuate rapidly among multiple types, so it becomes impossible for database administrators to manually adjust the system configurations in order to dynamically achieve performance objectives of the requests during runtime. Therefore, a*utonomic* workload management becomes necessary and critical to effectively control the process of requests and manage shared system resources to achieve required performance objectives in a complex request mix environment.

Since *autonomic computing* was first introduced [4], a great deal of effort has been put forth by researchers and engineers in both academia and industry to build autonomic computing systems. Autonomic computing systems (with the ability to self- configure, optimize, protect, and heal) are self-managing systems that manage their own behavior in accordance with high-level objectives specified by human administrators [4], [6]. Such systems regulate and maintain themselves without human intervention to reduce the complexity of system management and dynamically achieve system objectives, such as performance, availability and security objectives. In particular, an autonomic workload management system for DBMSs is a self-managing system that dynamically manages workloads present on a data server in accordance with specified high-level business objectives.

Achieving the goal of autonomic workload management involves using utility functions, a measure of preference of a choice, to map high-level business objectives to low-level DBMS tuning actions in order to guide a database system to optimize its own behavior and achieve required performance objectives. In this paper, we illustrate the use of utility functions in different aspects of workload management, namely dynamic resource allocation and query scheduling, to ensure mixed requests on a data server achieve their required performance objectives, and present a set of fundamental properties of a utility function used for building autonomic workload management systems. The primary objective of our research is to explore autonomic workload management for DBMSs. The contribution of this study is a method of evaluating utility functions in workload management systems for DBMSs as a step towards a self-managing system.

The paper is organized as follows. Section II reviews the background and related work of this study. Section III discusses fundamental properties of a utility function that can be used in autonomic workload management for DBMSs. Section IV presents two different types of utility functions defined in our studies. Finally, we conclude our work and propose future research in Section V.

II. BACKGOUND

In the past several years, considerable progress has been made in workload management for DBMSs. New techniques have been proposed by researchers, and new features of workload management facilities have been implemented in commercial DBMSs. These workload management facilities include IBM® DB2® Workload Manager and Query Patroller [5], Teradata® Active System Management [17], Microsoft® SQL Server Resource and Query Governor [11], [12] and Oracle® Database Resource Manager [16]. The workload management facilities manage complex workloads (that is, a mix of business processing and analyzing requests) present on a data server using predefined database procedures, based on the characteristics of the requests, such as estimated costs, resource demands and performance behavior. The predefined procedures use workload characterization, admission control, and execution control mechanisms to impose proper controls on the workloads to achieve their performance objectives.

Krompass *et al.* [9] and Metha *et al.* [13] presented a process of workload management for DBMSs that included three typical controls, namely admission, scheduling, and execution controls. Admission control determines whether or not an arriving request can be admitted into a database system, thus it can avoid increasing the load while the system is busy. Query scheduling determines the execution order of admitted queries and maintains the database system running in an optimal state. Execution control dynamically manages some running queries to limit their impact on other concurrently running requests.

Autonomic computing has been intensively studied in the past decade. Many autonomic computing components (with certain self-managing capabilities) have been developed and proven to be useful in their own right, although a large-scale fully autonomic computing system has not yet been realized [7], [14]. In particular, Tesauro *et al.* [18], [20] studied autonomic resource allocation among multiple applications based on optimizing the sum of utility for each application. In their work, a data center was used, which consisted of multiple and logically separated application environments (AEs). Each AE provided a distinct application service using a dedicated, but dynamically allocated, pool of servers, and each AE had its own service-level utility function specifying the utility to the data center from the environment as a function of some service metrics. The authors presented two methodologies, a queuing-theoretic performance model and model-free reinforcement learning, for estimating the utility of resources. They evaluated the two methodologies in the data center and highlighted tradeoffs between the two methods. Bennani *et al.* [2] presented another approach for the same resource allocation problems in the autonomic data center. In their approach, the predictive multi-class queuing network models were proposed to implement the service-level utility functions for each AE.

III. UTILITY FUNCTIONS IN WORKLOAD MANAGEMENT

Achieving autonomic workload management for DBMSs involves the use of utility functions. In this section we consider the following questions:

- Why are utility functions suited to autonomic workload management for DBMSs?
- What utility functions are good for building autonomic workload management systems?

The first question can be answered based on the research of Kephart *et al.* [8] and Walsh *et al.* [20], who proposed the use of utility functions to achieve self-managing systems. In the work, the authors presented utility functions as a general, principled and pragmatic way of representing and managing high-level objectives to guide the behavior of an autonomic computing system. Two types of policies were discussed in guiding behavior of a system, namely action policies and goal policies. An action policy is a low-level policy that is represented in the form of *IF (conditions) THEN (actions*). Namely, if some conditions are satisfied, then certain actions must be taken by the system. In contrast with an action policy, a goal policy only expresses high-level objectives of a system, and the system translates the high-level objectives into specific actions to every possible condition. Utility functions are proposed for the translation as it has the property of mapping system states to real numbers, and the largest number represents a system preferred state. In using utility functions, a computing system, via maximizing its utilities under each condition, recognizes what the goal states are, and then decides what actions it needs to take in order to reach those states. Thus by maximizing utilities, a computing system optimizes its own behavior and achieves the specified high-level objectives.

As introduced in Section I, in a mixed request data server environment, the concurrently running workloads can have different types, business importance levels, and performance objectives. These properties may dynamically change during runtime. It is impossible for human administrators to manually make an optimal resource allocation plan for all the workloads in order to meet their resource requirements, whereas, a utility function is suited for this situation, based on the properties discussed above. It dynamically identifies resource preferences for a workload during runtime, and all utility functions of the workloads can be further used to define an objective function. A solution to optimizing the objective function is an optimal resource allocation plan. Autonomic workload management systems use the resource allocation plan to allocate resources to the workloads and achieve the required performance objectives. Thus, to manage workloads in DBMSs, using utility functions is naturally a good choice.

To answer the second question, we start by discussing performance behavior of a workload. The performance of a running workload on a data server depends on the amount of desired system resources that the workload can access. Typically, the performance of a workload increases non-linearly with the amount of resources assigned to it. As an example, in executing an OLTP workload, by increasing the multi-programming levels, the throughput of the workload initially increases, but at a certain point the throughput starts to level off. That is, at the beginning when the workload starts to run with a certain amount of resource allocated, performance of the workload increases rapidly. However, with additional resources assigned to the workload, the performance increment of the workload becomes very small. This can be caused either by a bottleneck resource among the system resources, which significantly limits the workload performance increase, or it may be the case that the database system has become saturated.

Utility functions in database workload management must capture the performance characteristics of a workload and represent the trend of the changes in amount of assigned resources and the achieved performance. A utility function defined for database workload management should be a monotonically non-decreasing function, and is capable of mapping the performance achieved by a workload with a certain amount of allocated resources into a real number, *u*. There is no single way to define a utility function, and with certain practice or research requirements, utility functions can have widely varying properties. However, we believe the following properties are necessary for a proper utility function used in workload management for DBMSs:

- The value, *u*, should increase as the performance of a workload increases, and decreases otherwise.
- The amount of utility increase (*i.e.*, marginal utility) should be large as the performance of a workload increases quickly, while the marginal utility should be small as the performance of a workload increases slowly.
- The input of a utility function should be the amount of resources allocated to a workload or a function of the allocated resources, and the output, *u*, should be a real number without any unit.
- The value, *u*, should not increase as more resources are allocated to a workload when the workload has reached its performance objective.
- In allocating multiple resources to a workload, a utility function should be capable of identifying the critical resources for the workload.
- For objective function optimization, a utility function should have good mathematical properties, such as an existing second derivative.

The first two properties describe the general performance behavior of a workload, and the third property provides the domain and codomain for a utility function. These three properties are fundamental for a utility function that can be used in a workload management system for DBMSs. The fourth and fifth properties represent the relationships among workload performance, resource provisions and performance objectives. Namely, if a workload has met the performance objective, the utility function would inform the database system not to allocate more resources to the workload, and if there is a critical resource for a workload, the utility function would tell the system to provide the resource. The last property presents a way of effectively optimizing objective functions.

IV. Utility Function Applications

Two examples of the use of utility functions in workload management for DBMSs are presented in this section, namely dynamic resource allocation and query scheduling. The two utility functions are discussed with respect to the properties listed in Section III.

A. Dynamic Resource Allocation

In workload management for DBMSs, dynamic resource allocation can be triggered by workload reprioritization (a workload execution control approach [5], [21]). That means, a workload's business priority may be dynamically adjusted as it runs, thereby resulting in immediate resource reallocation to the workload according to the new priority.

Two shared resources are considered in the study, namely database buffer pool memory pages and system CPU shares. The DBMS concurrently runs multiple workloads, which are classified into different business importance classes with unique performance objectives. High importance workloads are assigned more resources, while low importance ones are assigned fewer. In the approach, all the workloads are assigned some virtual "wealth" to reflect the business importance levels. High importance workloads are assigned more wealth than low importance ones.

The dynamic resource allocation approach consists of three main components, namely a *resource model*, a *resource allocation method* and a *performance model*. The *resource model* is used to partition the resources and to determine a reasonable total amount of the resources for allocation. The *resource allocation method* determines how to obtain an optimal resource pair of buffer pool memory pages and CPU shares for a workload. In the approach, a greedy algorithm is used for identifying resource preferences of a workload. The resource allocation is determined iteratively. In an iteration of the algorithm, by using its virtual wealth, a workload bids for a unit of the resource (either buffer pool memory pages or CPU shares) that it predicts will yield the greatest benefit to its performance. The *performance model* predicts the performance of a workload with certain amount of allocated resources in order to determine the benefit of the resources. In the approach, queueing network models (QNM) [10] are used to predict the performance of a workload at each step of the algorithm. The details of this study are available elsewhere [21].

We consider OLTP workloads and use throughput as the performance metric to represent the performance required and achieved by the workloads. We model the DBMS used in our experiments for each workload with a single-class closed QNM, which consists of a CPU service center and an I/O service center. The CPU service center represents the system CPU resources and the I/O service center represents buffer pool and disk I/O resources. The request concurrency level of a workload in the DBMS is the number of database agents (CPU resources) assigned to the workload. The average CPU service demand of requests in the workload can be expressed as a function of the CPU shares allocated to the workload, using equation (1).

$$S_{CPU} = 1/(a * N + d) \qquad (1)$$

We experimentally defined the relationship between the CPU service demand and the number of database agents used in a DBMS. In the equation, N is database agents and $N \leq m$, and a and d as well as m are constants that can be determined through experimentation.

For an OLTP workload, the average I/O service demand can be expressed as a function of buffer pool memory size, which can be derived from *Belady's* equation [1]. The I/O service demand is:

$$S_{IO} = c * M^b \qquad (2)$$

where c and b are constants, and M is buffer pool memory pages assigned to the workload. In the equation the constants c and b can be determined through experimentation.

Performance of a workload with some allocated resources, <*cpu, mem*>, can be predicted by solving the analytical performance model with Mean Value Analysis (MVA) [10]. The predicted throughput of a workload can be expressed as a function of its allocated resources, using equation (3).

$$X = MVA(N,\ S_{CPU}(cpu),\ S_{IO}(mem),\ Z) \qquad (3)$$

where, X is the predicted throughput of a workload by using MVA on the QNM for a workload with its allocated resource pair, <*cpu, mem*>; N is the number of requests concurrently running in the system (*i.e.*, the number of database agents assigned to the workload); $S_{CPU}(cpu)$ is the average CPU service demand determined in equation (1); $S_{IO}(mem)$ is the average I/O service demand determined in equation (2); and Z is think time.

To guide workloads to capture appropriate resource pairs, utility functions are employed in the approach. We define a utility function that normalizes the predicted throughput from our performance model relative to the maximum throughput that the workload could achieve when all the resources are allocated to it. The utility function is given by:

$$u = MVA_{throughput}(N,\ S_{CPU}(cpu),\ S_{IO}(mem),\ Z)/X_{max} \qquad (4)$$

where, $MVA_{throughput}(w,\ x,\ y,\ z)$ is the predicted throughput determined in equation (3), and X_{max} is the maximum throughput achieved by a workload with all the resources allocated, which can be determined through experimentation.

This utility function is defined to map performance achieved by a workload with certain amount of resources into a real number u, $u \in [0, \dots, 1]$. If the utility of resources allocated to a workload is close to 1, it means the performance of the workload is high, while if the utility of resources allocated to a workload is close to 0, it means the performance of the workload is low. Workloads employ the utility function to calculate marginal utilities, that is, the difference in utilities between two possible consecutive resource allocations in a resource allocation process. As the utility function is non-decreasing, the value of a marginal utility is also in the range [0, …, 1].

The marginal utility reflects potential performance improvement of a workload. For some resources, if the calculated marginal utility of a workload is close to 1, then it means these additional resources can significantly benefit the workload's performance, while if the calculated marginal utility is close to 0, then the additional resources will not greatly improve the workload's performance. By examining the marginal utility value, a workload can determine the preferred resources for bid. The bid of a workload is the marginal utility multiplied by current available wealth of the workload, and indicates that a workload is willing to spend the marginal-utility percentage of its current wealth as a bid to purchase the resources. Wealthy workloads, therefore, can acquire more resources in the resource allocation processes and achieve better performance than poor ones.

B. *Query Scheduling*

Our *query scheduler* [14] is built on a DB2 DBMS and uses DB2 Query Patroller (DB2 QP) [5], a query management facility, to intercept arriving queries and acquire their information, and then determines an execution order of the queries. The *query scheduler* works in two main processes, namely workload detection and workload control. The workload detection process classifies arriving queries based on their service level objectives (SLOs), and the workload control process periodically generates new plans to respond to the changes in the mix of arriving requests.

We consider a system with n service classes, each with a performance goal and a business importance level, denoted as $\langle \overline{g}_i, m_i \rangle$, where $\overline{g}_i$ is the performance goal of the *i-th* service class, and m_i is the class business importance level. The pair $\langle \overline{g}_i, m_i \rangle$ is a service level objective. We denote $g_1, g_2, \cdots, g_n$ as the predicted performance of the n service classes given a resource allocation plan $r_1, r_2, \cdots, r_n$ (multi-programming levels in our case). The performance of the *i-th* service class, g_i, can be predicted by using a performance model given r_i, the amount of resources allocated to the service class. The utility of the *i-th* service class, u_i, can be expressed as a function of $\overline{g}_i$, m_i and g_i, namely $u_i = f_i(\overline{g}_i, m_i, g_i)$, and the n SLOs can be encapsulated into an objective function f, $f(u_1, u_2, \cdots, u_n)$. Thus, the scheduling problem can be solved by optimizing the objective function f. By properly defining the functions of f_i 's and f, an optimal resource allocation plan can be derived by maximizing the objective function.

In the study, we specifically consider business analysis requests, such as those found in data warehousing decision support systems. The TPC-H benchmark [19] is used as the database and workloads in our experiments. We apply the performance metric - *query execution velocity*, the ratio of expected execution time of a query to the actual time the query spent in the system (the total time of execution and delay), to represent the performance required and achieved by the queries, since queries in decision support systems can widely vary in their response times. In using query execution velocity, performance goals as well as business importance levels of queries can be well captured.

Based on the experiments we found the following general form of utility functions satisfies our requirements:

$$u = 1 - e^{\alpha m \frac{\bar{g}-g}{g-\hat{g}}} \tag{5}$$

where, $\bar{g}$ is the performance goal of a service class to be achieved, m is the importance level of the service class, $\hat{g}$ is the lowest performance allowed for the service class, g is the actual performance, and α is an importance factor that is a constant and can be experimentally determined or adjusted to reflect the distance between two adjacent importance levels. In using α, we control the size and shape of the utility function.

The objective function, *f*, is then defined as a sum of the service class utility functions, using equation (6):

$$f = \sum_{i=1}^{n} u_i \tag{6}$$

In *query scheduler*, the *performance solver* employs a performance model to predict query execution velocity for a service class. That is, given a new value of service class cost limit, the performance of the service class can be predicted for the next control interval, which is based on its performance and service class cost limit at the current control interval. The performance at the next control interval is predicted by:

$$V_i^k = \begin{cases} V_i^{k-1} C_i^k / C_i^{k-1} & if\ V_i^{k-1} C_i^k / C_i^{k-1} \le 1 \\ 1 & if\ V_i^{k-1} C_i^k / C_i^{k-1} > 1 \end{cases} \tag{7}$$

where, V_i^{k-1} and V_i^k are query execution velocity of service class *i* at *(k-1)-th* and *k-th* control intervals, respectively; C_i^{k-1} and C_i^k are cost limits of service class *i* at the *(k-1)-th* and the *k-th* control intervals, respectively.

Therefore, a scheduling plan can be determined. From equations (5), (6) and (7), we have:

$$f = \sum_i^n u_i^k \tag{8}$$

$$u_i^k = 1 - e^{a_i m_i \frac{\bar{V}_i - V_i^k}{V_i^k - \hat{V}_i}} \tag{9}$$

$$V_i^k = V_i^{k-1} C_i^k / C_i^{k-1} \tag{10}$$

replacing V_i^k in equation (9) with equation (10) and u_i^k in equation (8) with equation (9), the solution for maximizing the objective function, $f(C_1^k, C_2^k, \cdots, C_n^k)$, is the query scheduling plan for *k-th* control interval, where the object function must maintain the constraint, $C_1^k + C_2^k + \cdots + C_n^k \le C$, and C is the system cost limits.

C. Discussion

In these two examples, for dynamic resource allocation, the utility function was defined based on a single-class closed QNM, while, for query scheduling, the utility function was chosen based on an exponential function. These two types of utility functions are different in their forms and research requirements, but both strictly maintain the same fundamental properties listed in Section III. The input of the dynamic resource allocation utility function is an amount of allocated resources (the resource pair, <*cpu, mem*>), the output is a real number in the range [0, 1], and the applied QNM properly predicts performance behavior of the workload. The input of the query scheduling utility functions is a performance model with allocated resources (the query execution velocity of a service class, as described in Subsection *B*) and the output is a real number in (-∞, +∞). Based on the exponential function properties, as the input of the utility function increases, the output (utility) increases and, at a certain value, it begins to level off. That means, when the service class approaches its performance goal, the utility increase is less, and it indicates that the database system should not assign more resources to the service class.

For a defined objective function, if it is continuous, the *Lagrange* method can be applied to solve it, otherwise searching techniques may be used. In query scheduling, the defined utility function has the second derivative existing, and this allows mathematical methods to be applied to optimize the objective function.

In evaluating the two types of utility functions based on the set of properties listed in Section III, both utility functions preserve the fundamental properties, that is, a) the utility increases as a workload performance increases, and decreases otherwise; b) the marginal utility is large as a workload performance increases quickly, and is small otherwise; c) the input and output are in the required types and values. In comparing the two utility functions presented in Table 1, we observe that the utility function used in dynamic resource allocation has the property of identifying critical resources for a workload, but it does not have mathematical properties for optimizing an objective function. The utility function used in query scheduling possesses a good mathematical property for optimizing its objective function, but it does not have the property of identifying system critical resources.

Since the utility functions were strictly defined based on their research requirements, the specific research problems shaped the utility function's properties. So, we conclude that the two types of utility functions are good in terms of their specific research requirements and considered acceptable based on the set of properties listed in Section III.

TABLE I. COMPARISION OF THE TWO UTILITY FUNCTIONS

	Utility functions in Dynamic Resource Allocation	**Utility functions in Query Scheduling**
Utility Increasing Normally	yes	yes
Marginal Utility Increasing Normally	yes	yes
Utility Function Input	allocated resources	a function of the allocated resources
Utility Function Output	a number in [0, 1]	a number in **(**-∞, +∞**)**
Critical Resource Identifying	yes	no
Having Mathematical Property	no	yes
Utility Increase Stops as Goals Achieved	yes	yes

V. Conclusion And Future Work

In this paper we have presented two concrete examples to illustrate how utility functions can be applied to database workload management, namely dynamic resource allocation and query scheduling. Based on the examples, we generalized a set of function properties that is fundamental for defining utility functions in building autonomic workload management for DBMSs in future practice and research.

As more workload management techniques are proposed and developed, we plan to investigate the use of utility functions to choose during runtime an appropriate workload management technique for a large-scale autonomic workload management system, which can contain multiple techniques. Thus, the system can decide what technique is most effective for a particular workload executing on the DBMS under certain particular circumstance.

Acknowledgment

This research is supported by IBM Centre for Advanced Studies (CAS), IBM Toronto Software Lab, IBM Canada Ltd., and Natural Science and Engineering Research Council (NSERC) of Canada.

Trademarks

Disclaimer

References

[1] L. A. Belady. “A Study of Replacement Algorithms for a Virtual-Storage Computer”. IBM Systems Journal, Volume 5, Issue 2, June 1966, pp. 78-101.

[2] M. N. Bennani and D. A. Menasce, “Resource Allocation for Autonomic Data Centers using Analytic Performance Models”, In Proc. of the Intl. Conf. on Autonomic Computing, (ICAC’05), Seattle, Washington, USA, 13-16 June, 2005, pp. 229-240.

[3] D. P. Brown, A. Richards, R. Zeehandelaar and D. Galeazzi, “Teradata Active System Management: High-Level Architecture Overview”, A White Paper of Teradata, 2007.

[4] IBM Corp., “Autonomic Computing: IBM’s Perspective on the State of Information Technology”. On-line Documents, retrived in February 2011. http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.

[5] IBM Corp., “IBM DB2 Database for Linux, UNIX, and Windows Information Center”. On-line Documents, retrived in Feb. 2011. https://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

[6] J. O. Kephart and D. M. Chess, “The Vision of Autonomic Computing”, Computer, Volume 36, Issue 1, January 2003, pp. 41-50.

[7] J. O. Kephart, “Research Challenges of Autonomic Computing”. In Proc. of the 27th Intl. Conf. on Software Engineering (ICSE’05). St. Louis, MO, USA, 15-21 May, 2005, pp. 15-22.

[8] J. O. Kephart and R. Das, “Achieving Self-Management via Utility Functions,” IEEE Internet Computing, Vol. 11, Issue 1, January/February, 2007, pp. 40-48.

[9] S. Krompass, H. Kuno, J. L. Wiener, K. Wilkison, U. Dayal and A. Kemper, “Managing Long-Running Queries”, In Proc. of the 12th Intl. Conf. on Extending Database Technology: Advances in Database Technology (EDBT’09), Saint Petersburg, Russia, 2009, pp. 132-143.

[10] E. Lazowska, J. Zahorjan, G. S. Graham and K. C. Sevcik “Quantitative System Performance: Computer System Analysis Using Queueing Network Models”, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1984.

[11] Microsoft Corp., “Managing SQL Server Workloads with Resource Governor”. On-line Documents, retrived in February 2011. http://msdn.microsoft.com/en-us/library/bb933866.aspx.

[12] Microsoft Corp., “Query Governor Cost Limit Option”, On-line Documents, retrived in February 2011. http://msdn.microsoft.com/en-us/library/ms190419.aspx.

[13] A. Mehta, C. Gupta and U. Dayal, “BI Batch Manager: A System for Managing Batch Workloads on Enterprise Data-Warehouses”, In Proc. of the 11th Intl. Conf. on Extending Database Technology: Advances in Database Technology (EDBT’08). Nantes, France, March 25-30, 2008, pp. 640-651.

[14] D. A. Menasce and J. O. Kephart, “Guest Editors' Introduction: Autonomic Computing”, In IEEE Internet Computing, Volume 11, Issue 1, January 2007, pp. 18-21.

[15] B. Niu, P. Martin and W. Powley, “Towards Autonomic Workload Management in DBMSs”, In Journal of Database Management, Volume 20, Issue 3, 2009, pp. 1-17.

[16] Oracle Corp., “Oracle Database Resource Manager”, On-line Documents, retrived in February 2011. http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/dbrm.htm#i1010776.

[17] Teradata Corp., “Teradata Dynamic Workload Manager”, On-line Documents, retrived in February 2011. http://www.info.teradata.com/templates/eSrchResults.cfm?prodline=&txtpid=&txtrelno=&txtttlkywrd=tdwm&rdsort=Title&srtord=Asc&nm=Teradata+Dynamic+Workload+Manager.

[18] G. Tesauro, R. Das, W. E. Walsh and J. O. Kephart, "Utility-Function-Driven Resource Allocation in Autonomic Systems", In Proc. of the 2nd Intl. Conf. on Autonomic Computing (ICAC’05), Seattle, Washington, USA, 13-16 June, 2005, pp.342-343.

[19] Transaction Processing Performance Council. http://www.tpc.org.

[20] W. E. Walsh, G. Tesauro, J. O. Kephart and R. Das. “Utility Functions in Autonomic Systems”, In Proc. of the 1st Intl. Conf. on Autonomic Computing (ICAC’04), New York, USA, 17-18 May, 2004, pp.70-77.

[21] M. Zhang, P. Martin, W. Powley and P. Bird. “Using Economic Models to Allocate Resources in Database Management Systems”, In Proc. of the 2008 Conf. of the Center for Advanced Studies on Collaborative Research (CASCON’08), Toronto, Canada, Oct. 2008, pp. 248-259.

A Framework for Certifying Autonomic Computing Systems

Haffiz Shuaib, Richard J Anthony and Mariusz Pelc
School of Computing and Mathematical Sciences
The University of Greenwich
Park Row, Greenwich, London SE10 9LS, UK
Email: {H.Shuaib,R.J.Anthony,M.Pelc}@gre.ac.uk

***Abstract*—The growing cost associated with managing increasingly complex Information Technology (IT) infrastructure has brought about a new field of research – Autonomic Computing (AC). AC systems seek to mimic the management capabilities of biological systems notably, the Autonomous Nervous System (ANS), to bend the curve associated with management cost, while also allowing these managed systems to evolve and adapt, regardless of the operational context and deployment environment. The AC field attempts to bring together a number of disparate research fields in order to achieve its objectives. For this to be possible, a way to coherently link the imports of these dissimilar fields is required, if these systems are to be understood, and by extension trusted. To that end, this project's focus is on the proposal of mechanisms that will allow for proper certification of autonomic computing systems. This, if successful, will provide a consistent trust measure for these systems. Particularly, in this position paper the intelligent machine design architecture is extended and used as a basis for defining autonomic computing systems. Metrics contained in the International Standard Organization (ISO) 9126-1998 specification, are appropriated to AC systems. Based on all of the above, a foundation for achieving certification for AC systems is laid.**

***Keywords*-Autonomic Computing, Architectures, Frameworks.**

I. Introduction

A true Autonomic Computing (AC) system is one that is able to automate the management decision-making process and reflect on the quality of the decisions made. This it must do regardless of the environmental context and within the goals set by the human operator. The ultimate aim of autonomic computing systems is to allow complex Information Technology (IT) infrastructure evolve into more complex systems to handle more difficult tasks, without significantly increasing the cost of management.

The ability to self-configure, self-optimize, self-protect and self-heal, have been identified as the four cardinal characteristics of AC systems [1]. As with most critical or increasingly complex systems, an AC system should and must be certified on the basis of its expected characteristics before it goes live, as these systems have implications from the financial to the space exploration industries. Achieving this system certification task is made more challenging still, given that the AC field itself draws from a disparate number of well established fields, including artificial intelligence, telecommunications research, mathematics, software/hardware engineering, statistics etc. each with its own view of how a system is to be defined.

No attempt has been made towards the certification of autonomic computing systems thus far; therefore, the primary objective of this project is the provision of a technical and visible support structure for the relevant components of these disparate fields, to facilitate the certification process of AC systems. It is hoped that the final result of this project will provide a basis; for assessing autonomic systems with similar functionalities, for assessing the current capability of the system and its suitability to the problem, to assess the impact of a certified component on a system and to resolve legal liability, if the autonomic computing systems were to fail.

The rest of this paper is organized as follows; the operational mechanisms of AC system are expected to mimic those of the biological Autonomic Nervous System (ANS), relevant aspects of the ANS are discussed in the next section. The state of the art as it relates to AC system architecture is discussed in Section III. Also presented in this section, is a proposed extension of the Intelligent Machine Design as a basis for AC system architecture. In Section IV, the challenges associated with the certification of AC systems are discussed. Further discussed are suggested metrics by which, compliant AC systems can be measured. Section V contains the conclusion and future work.

II. Autonomic Nervous System

The Autonomous nervous system (ANS) in part inspires the Autonomic Computing (AC) field; as a result, it is pertinent that it is discussed briefly in this paper. This discussion will be restricted to those components of the ANS that are believed to be relevant to AC systems.

The ANS relies on three major components to achieve its independent management objectives; Neurons, synapses, inter-neuron communication protocols and excitatory and inhibitory mechanism that modulate the output or response of the ANS for a given input signal. The Neurons are the processing units and the basic building blocks of the ANS. These neurons are connected to one another by directional links called synapses. Information is conveyed through the synapses in the form of Ions, which, have a similar external structure but may differ in internal chemical composition. The excitatory and inhibitory mechanism modulates the output or response of the ANS for a given input signal. In effect, regardless of the magnitude of the input signal, the output response can be graded to meet an overall objective.

A few salient concepts from the ANS that apply to ACs are as follows: A standard communication channel, similar to the ANS synapse that can carry information to and from the AC's processing entities or the AC's equivalent of neurons. This is especially important, as these entities may be spread over several physical locations or even spread over a number of disparate hardware and software platforms. Recall, that the synapses convey information in the form of ions, which, internally may differ in terms of chemical composition. A dynamic description language is required in this regard. As will be shown later in this work, a policy definition language based on the Extensible Markup Language (XML) is appropriate for this. A means to have an all or nothing response or a graded response based on the input information from the sensory mechanism(s) and acquired knowledge of the systems is also required. A means to marshal resources to areas most in need in an emergent scenario and back to a steady state is also desirable. These concepts when implemented technically must be portable and platform agnostic.

III. A Reference Architecture for an Autonomic Computing Element

An architectural standard is central to the process of the certification of a system. Any architecture that represents an autonomic computing system must not be narrowly defined such that it precludes the ability for the system to evolve or cater for new use cases. As noted in [2] and [3], the lack of an open standard is a challenge in the autonomic computing field. In this section, the most prevalent of all autonomic architectures i.e., IBM's MAPE architecture is discussed. Drawbacks relating to this architecture are also discussed, and a certifiable alternative architecture with similar functionalities is presented.

Before proceeding to the next sub-section a few definitions are in order;

Definition 1: **Autonomic Manager:** This component independently makes decisions that are then effected on a managed component.

Definition 2: **Autonomic Element:** This consists of both the management (autonomic manager) and managed components.

Definition 3: **Goal:** This is the overall objective of the autonomic system. For instance, the overall goal of a vehicle might be to get from a point A to a Point B. How this is achieved is left to low-level functionalities.

Definition 4: **Rule:** A rule is a ***Boolean*** statement that evaluates to true or false, with an action taken or not taken based on the outcome [4].

Definition 5: **Policy:** Finally, a policy is made up of a number of rules. The primary objective of policies is to achieve the overall system goal in the most efficient way possible [4].

A. IBM's Autonomic Architecture

The well known IBM MAPE (Monitor, Analyze, Plan, Execute) AC architecture consists of four main components which, form a loop, as shown in Figure 1.The first of these components is the Monitor. Its main duty is to monitor the surrounding environment, including system resources. The output of this Monitor is used for making decisions at later stages of the loop. The second component i.e., the Analyze component, uses a number of algorithms to anticipate problems and possibly proffer solutions to these problems. The Planning component uses the information available to the autonomic system to choose which, policies to execute. The Execution component, which, is the fourth component, effects the most appropriate policy/policies chosen by the system. This executed policy may cause a change in the physical environment e.g., moving the arm of a robot, or simply pass instructions or information to another element, possibly an autonomic one. The input to the MAPE architecture comes from the sensory mechanism, while the effector mechanisms carry out the dictates of the machine.

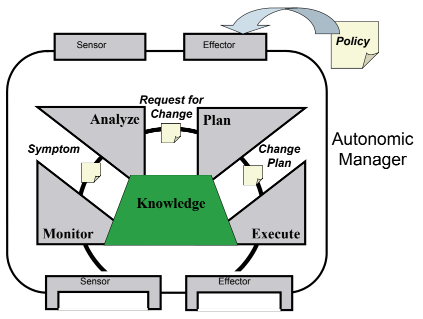


Figure 1. IBM Autonomic MAPE Architecture [1]

The functionality associated with any one of these component can be delegated wholly to the human manager or the autonomic system. The level of dependence on the machine or the human operator is calibrated using a metric called the Autonomic Maturity Index (AMI).

IBM defines five indices in this regard [1]. They are as follows;

- Manual (Level 1): At this level, the human operator is completely responsible for all functional components of the MAPE architecture.
- Instrument and monitor (Level 2): Here, the autonomic system is responsible for the collection of information (Monitoring). This collected/aggregated information is analyzed by the human operator and guides future actions of the operator.
- Analysis (Level 3): In this level, information is collected and analyzed by the system. This analyzed data is passed to the human administrator for further actions.
- Closed loop (Level 4): This works in the same way as the Analysis level, only this time the system's dependence on the human is minimized i.e., the system is allowed to action certain policies.
- Closed loop with business processes (Level 5): At this level, the input of the administrator is restricted to creating and altering business policies and objectives. The system will operate independently using these objectives and policies as a guide.

In the early days of the AMI i.e., 2003, it was believed that most computing systems resided at the Basic level [5]. However, research prototypes conforming to higher levels are said to exist currently [6].

There is no consensus on whether the IBM MAPE architecture is a concrete architecture or a malleable concept. For instance, [7] defines the architecture as canon and then goes on to implement it to the letter. [8] and [9] both assume MAPE to be a concrete architecture that can be tweaked. For example, [8] collapses the four main components of the MAPE architecture into two groups. The Monitor/Analyze components are placed into one group, with this group given the responsibility of handling issues like fault diagnoses and anomaly detection. The Plan/Execute group deals with issues relating to resource-allocation and configuration. [9], in a similar manner to [8] breaks the architecture into two main groups, but this time the Monitor/Analyze components handle tasks that are reactive, while the Plan/Execute components are responsible for proactive adaptation. [10] adopts a slightly different approach to modifying the architecture. The authors of [10] divide the architecture into a global and local sub-architecture, with the Analyze/Planning and Monitor/Execute components implemented in the global and local sub-architectures, respectively.

As a concept, the IBM architecture is said to require concrete expressions for it to be applicable to the framework discussed in [11]. The MOSES framework [12] which, seeks to manage quality of service expectations in a volatile environment also falls under the concept school of thought. It maps each component of the MAPE architecture to its own architecture. [13] proposes to make the MAPE architecture real or concrete by generalizing the model for large-scale data processing infrastructures. In [14], the MAPE architecture is seen as a concept that can be applied to an architecture, not as an architecture itself.

Beyond the concept/concrete architecture argument, some researchers have stated that the MAPE architecture is flawed, or at least insufficient to describe autonomic systems. For example, [15] consider the architecture concrete, but too narrowly defined to apply to some autonomic systems e.g., multi-agent systems. [16] points out that the loop in the MAPE architecture is vulnerable to failure, which, in turn can precipitate the collapse of the management system all together. A solution to this problem might be a situation where an outer loop monitors the internal loop as was done in [8]. However, this begs the question, is another outer, outer loop needed to also monitor the outer loop and so on and so forth? Rather than using external and rigid feedback loops to make the MAPE loop more resilient, the authors of [16] propose the addition and the removal of loops on the fly. Their solution is inspired by biological emergent systems described in [17]. [18] also adds a partial outer loop to the MAPE architecture to allow the system to evolve.

Regardless of its wide applicability, the divergent views associated with the IBM MAPE architecture makes it ill-suited for the certification of autonomic systems.

The IBM's AMI, while critical to the certification process, is said to be narrowly defined and technically vague [5]. This makes it difficult to align an autonomic system with these maturity indices [19]. Obviously, these concerns do not help the certification process. An attempt is made in the next section to address the above. It is pertinent to mention here that several other forms of accessing the level of autonomicity have also been proposed in the past, notably those in [20]. Here, the level of autonomicity is defined by who makes the decisions and how these decisions are executed [21]. From a certification perspective, the position of a system on the autonomic maturity index should be defined by who makes the decisions and the quality of the decisions themselves. Both metrics will engender a certain level of trust in the system.

B. An Alternative Architecture

The appeal of the Intelligent Machine Design (IMD) architecture [22] to autonomic computing systems is that it is closely related to the way intelligent biological systems work. Indeed, this architecture has been suggested as a generic framework on which, autonomic systems can be built upon [23]. While this architecture is mentioned in some autonomic computing literature, nothing concrete from a technical perspective has been achieved relative to IBM's architecture (See Figure 2). In this section, an attempt is made to imbue this architecture with the self-management

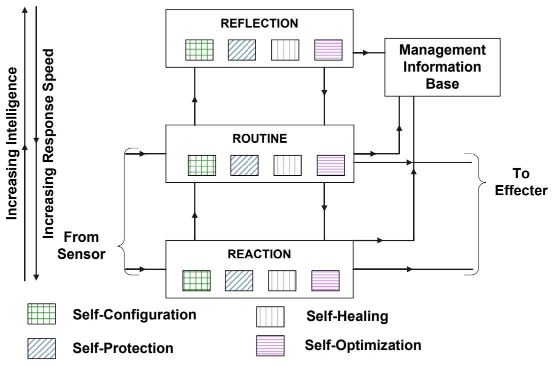


Figure 2. An Autonomic Computing expression of the IMD

autonomic properties and further relate it to a proposed AMI. Before going into the technical details, a quick overview of the make up of the architecture is presented.

The architecture supposes that an intelligent machine is made up of three distinct layers, i.e., Reaction, Routine and the Reflection level. Each layer can be characterized by the following attributes; the amount of resources consumed, their ability to activate/inhibit the functionality of a connected layer and their ability to be activated or inhibited by another layer. Attributes that deal with the preoccupation of each level are also added later in this section.

The lowest layer, the Reaction layer, is connected to the sensors and effectors. When it receives a sensory stimulus, it responds relatively faster than the other two layers. The primary reason for this is that its internal mechanisms are simple, direct and hardwired i.e., it has an automatic response to incoming signals. A simple IF-ELSE statement suffices here. In autonomic computing parlance, the human operator does much of the learning or monitoring/analysis. The Reaction layer takes precedence over all other layers and can trigger higher layer processing. This layer can also be inhibited/activated by the Routine layer. It consumes the least amount of resources. It might also have a lmited access to the working memory or the management information base (MIB).

The Routine (mid-level) layer is more learned and skilled when compared to the Reaction layer. It is expected to have access to the working memory (or MIB), which, contains a number of policy definitions that can be executed based on context, knowledge and self-awareness. As a result, it is comparatively slower than the Reaction level. Its activities can be activated or inhibited by the Reflection layer. Its input comes from both the sensory mechanism and the Reflection layer. Its output goes to the effector mechanism and the Reflection layer. When the Routine level is unable to find a suitable policy for an immediate objective, it hands control over to the human administrator or the Reflection layer.

While the Routine level's primary objective is to deal with expected situations whether learned or hardwired, the Reflection level, which, is the highest level, helps the machine deal with deviations from the norm. The Reflection level is able to deal with abnormal situations, using a combination of learning technologies (e.g., Artificial Neural Networks, genetic algorithms), partial reasoning algorithms (e.g., Fuzzy Logic, Bayesian reasoning), the machine's knowledge base, context and self-awareness. Technically, the Reflection Layer's ultimate aim, as it relates to autonomic computing systems, is to create and validate new policies at runtime that will be used at the Routine level. If the system is able to adapt to an unexpected situation as a result of the new policy, then the policy is stored in the working memory (or MIB). This new policy can be called upon if the situation is encountered in the future. Thus, making a formerly abnormal situation a routine one. The process of ’reasoning’ out a new policy makes the Reflection layer the largest consumer of computing resources. This also means it has the slowest response time of all three layers. The Routine layer is the input source and output destination for the Reflection layer. The Reflection layer can inhibit/activate the processes of the Routine layer through new policy definitions. Higher layers are able to excite or inhibit the activities of the lower layers in a manner similar to how the ANS is modulated by knowledge or the conscious mind. Notice from Figure 2 that all layers will be able to implement the four cardinal self-management properties. Also, in the same way in which, the ANS communicates between neurons using ions irrespective of their chemical composition, autonomic computing elements implementing the architecture shown in Figure 2 should also be able to communicate with one another using a standard mechanism. In addition, the old and new policies must use a standard definition language to ensure consistency across the board. The policy framework defined in RFC 3060/3460 is the vehicle by which, the above is to be achieved.

C. *The Alternative Architecture and the AMI*

The architecture shown in Figure 2 can be associated with the AMI. To do this, an attempt is made to expressly define what each Maturity Index means from a technical perspective, and further relate each index to the layers of the machine. The Five maturity indices are thus interpreted as;

- **Maturity Index 1:** Here, only one policy action is executed in response to all input signals and encountered contexts. Complex operations are referred to the human operator or to the immediate higher level. This maturity index corresponds to the Reaction level.
- **Maturity Index 2:** This index corresponds to the Routine level. If the Routine level is unable to find a suitable policy from a policy repository or if there is a policy ambiguity, it relies on the human administrator to provide a new solution or resolve the policy conflict.
- **Maturity Index 3:** This is similar to Maturity Index 2, only that this time, the Routine layer consults the Reflection layer to solve its policy problems.
- **Maturity Index 4:** This index corresponds to the Reflection layer. The Reflection layer of a Machine in this index will attempt to solve the policy problem of the Routine layer, and monitor the implementation of this new policy. If the policy fails in its objective or if a new policy cannot be created, the human administrator is required to intervene.
- **Maturity Index 5:** This is similar to index 4, but rather than defer to the human administrator, if a suitable policy is not found or created, the algorithm within the Reflection layer will continually attempt to create a new policy or resolve the policy conflict. This index should be used to define autonomic machines that will be unable to get in touch with the human manager, a craft in deep space for example. Another possible example for this index is a scenario where the human intervention cannot be timely enough due to the complexities in the system.

In effect, the autonomic maturity level 1 corresponds to the Reaction layer, levels 2 and 3 correspond to the Routine layer, 4 and 5 correspond to the Reflection layer. The position of an autonomic computing system on the defined maturity indices above provides a possible basis for verifying the source of the decision making process and the quality of the decisions made. For instance, if a system in question specifies a Maturity Index of 2, the certification process would know that the ’court of last instance’ is the human administrator. The certification process would now seek to verify the qualification of skilled personnel for the system to be awarded an index of 2. If the system seeks to be tagged with an index of 5 i.e., the decision making process is handled ultimately by the machine itself, the algorithm implemented in the Reflection layer must be shown to be robust enough to handle this task. Recall, that it was said that the source and quality of the decisions made is central to the trust placed on the system, and as a result relevant to the certification process (See the last paragraph of Section III-A).

IV. Verifying, Validating and Certifying Autonomic Computing Systems

The ultimate goal of this project is to define formal mechanisms, where possible, by which, autonomic systems can be certified. The road to certification, of course, is a long one that entails building, verifying and validating the system(s) in question. Before discussing the challenges associated with certifying autonomic systems, it is pertinent to define what is meant by validation, verification and certification. According to the IEEE standard glossary for Software Engineering terms [24], the ***Validation*** process seeks assurances that the system has been built as per the initial requirements set out. In other words, is the end-product fit for the purpose for which, it is built. The purpose of the ***Verification*** process is to ensure that the system performs correctly and consistently in terms of the input and output. ***Certification*** is defined as a written guarantee, a formal demonstration or the process of confirming that an offered component complies with specified requirements, and will work as expected in a defined environment. Clearly, certification encompasses the verification and validation processes.

The validation and verification techniques used for many large-scale software projects will suffice for some aspects of the autonomic systems, still other areas will require new constructs. For instance, classical testing techniques such as dynamic, static testing, formal methods will be well suited to levels 1 and 2 of the Autonomic maturity index defined in Section III-C. Nevertheless, these techniques are unsuitable for levels 3-5. The primary reason for this is that the behaviour of the system is expected to adapt, and change over time due to the activities of the Reflection layer. This implies that a means for these systems to self-validate within the context of the defined goal is required. One way to go about verifying/validating these systems is by separating the architecture from the algorithm contained within.This is similar to what was done in the DySCAS project [25], in which, a middleware to support self-configuring automotive systems was developed. Here the autonomic decision-making logic was developed and validated independently of the actual middleware mechanisms. For example, if the Reflection layer of a product implements an artificial neural

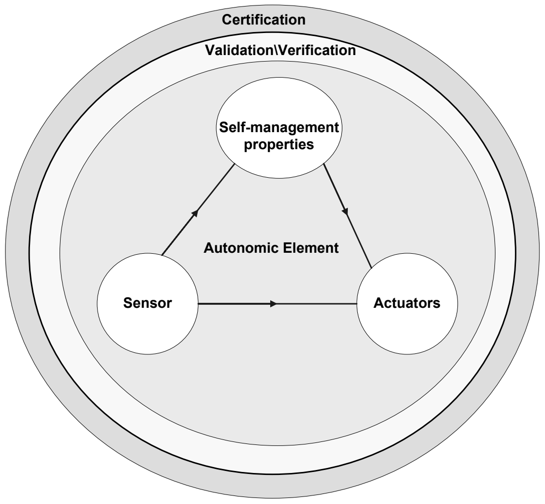


Figure 3. Verification, Validation and Certification

network algorithm, one can first check to see if the architecture implemented in the product meets a certain criteria e.g., portability, stability etc. using already established methods. If the architecture meets expectations, the arduous task of validating and verifying the algorithm follows. Otherwise, if the implemented architecture fails to meet expectations, the product should be deemed to have failed the test, and therefore there is no need to proceed to validate the algorithm implemented within. Validating/Verifying the algorithm e.g., artificial neural networks or genetic algorithms, on the other hand would require new formal methods of verification/validation or at least an extension of the known techniques. Some research efforts have been expended in this regard. Jacklin et al. [26] propose that any testing method for artificial neural networks, whether new or modified must take cognizance of the following; the level of detail and the correctness of the training data, the impact of data outside the training set on the system, the amount of memory available to hold the data set being operated upon, and the impact of the network algorithm on associated components. Needless to say, a strong grasp of the fundamentals of these learning algorithms is mandatory for this exercise.

A crucial aspect of correctly assessing the quality of an autonomic computing system is knowing what to measure and where to take these measurements. This task is often very difficult as highlighted in [27]. Several metrics have been identified, using the ISO 9126-1998 standard as a guide [28], that can be measured within the context of the four cardinal self-* properties of autonomic systems i.e., self-configuration, self-protection, self-healing and self-optimization. [28] defines six main characteristics that could be used to assess the quality of a software product, including; Functionality, Usability, Portability, Reliability, Efficiency and Maintainability. The Usability, Reliability and Maintainability attributes are implicit in the definition of autonomic systems, because of this, only the attributes relating to Functionality, Portability and Efficiency will be considered here.

The Functionality attribute consists of the following characteristics; suitability to the problem and interoperability. The speed of response, processing times and throughput of the system are characteristics of the Efficiency attribute. Finally, the Portability component consists of adaptability, installability, co-existence and replaceability. All have a sub-attribute that deals with compliance to standards. These three main attributes can and should be applied to each of the four main self-* properties during the validation, verification and certification process.

V. Conclusion and Future Work

Relative to currently deployed IT systems, autonomic computing system are expected to exhibit superior control/management behaviour and high adaptability, regardless of the operational context. However, a means for measuring this superiority is lacking. In this paper, two areas that allow for these assessments were tackled i.e., the architecture and the metrics that will allow for certification.

In this paper, an evaluation of the way in which, MAPE is not well-suited for certifiable systems was provided. The intelligent machine design architecture was discussed, and shown to be amenable to certification. Five technical autonomic maturity indices, that are similar in broad strokes to those proposed by IBM, were also presented and aligned to the intelligent machine design architecture. As it relates to metrics of autonomic systems, three of the six metrics defined in ISO 9126-1998 i.e., Functionality, Efficiency and Portability, were shown to be relevant to assessing autonomic systems, and are under ongoing investigation.

The work presented in this paper has laid the foundation on which, the certification of autonomic systems can be pursued. However, there is still much work to be done. The next tasks include;

- Firmly establishing the intelligent machine design architecture for autonomic systems such that it enforces structure but does not restrict the self-management algorithms implemented within. Clearly defining interfaces between any two autonomic elements or managers, or between any two levels on the architecture.
- Propose and implement mechanisms that will allow for efficient management coordination among elements of an autonomic system. These mechanisms will allow for proper establishment of administrative relationships between elements, will provide components that allow for management conflict resolution, if two or more autonomic managers simultaneously effect changes on a single managed system etc.
- Define and demonstrate steps by which autonomic computing systems are to be rated within the context of the targeted application domain. In the first instance, qualitative measures that fully describe the architectural make up of each manager and their associated maturity indices will be proposed. These qualitative metrics in turn will point to what quantitative measures or performance characteristics can be obtained from the machine under an evaluation scenario. As discussed previously, the quantitative measures will be based on ISO 9126-1998.

References

[1] IBM, *An architectural blueprint for autonomic computing.* IBM Corporation, 4 ed., June 2006.

[2] M. Salehie and L. Tahvildari, "Autonomic computing: Emerging trends and open problems," *DEAS'05. Workshop on the Design and Evolution of Autonomic Application Software*, vol. 30, pp. 1–7, 2005.

[3] R. Sterritt, "Autonomic computing," *Innovations System Software Engineering (2005), Springer-Verlag*, vol. 1, no. 1, pp. 79–88, 2005.

[4] R. J. Anthony, "A policy-definition language and prototype implementation library for policy-based autonomic systems," *ICAC '06. IEEE International Conference on Autonomic Computing*, pp. 265 – 276, 2006.

[5] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing degrees, models, and applications," *ACM Computing Surveys*, vol. 40(3), August 2008.

[6] B. Dillenseger, T. Coupaye, M. Salaun, and M. J. Barros, "Autonomic computing and networking:the operators' vision on technologies, opportunities, risks and adoption roadmap," *Eurescom study report*, p. 21, 2009.

[7] C. Reich, K. Bubendorfer, and R. Buyya, "An autonomic peer-to-peer architecture for hosting stateful web services," *CCGRID '08. 8th IEEE International Symposium on Cluster Computing and the Grid*, 2008.

[8] C. Kennedy, "Decentralised metacognition in context-aware autonomic systems: some key challenges," *In American Institute of Aeronautics and Astronautics (AIAA) AAAI-10 Workshop on Metacognition for Robust Social Systems, Atlanta, Georgia*, 2010.

[9] P. de Grandis and G. Valetto, "Elicitation and utilization of utility functions for the self-assessment of autonomic applications," *ICAC '09. International Conference on Autonomic Computing*, 2009.

[10] C. Dorn, D. Schall, and S. Dustdar, "A model and algorithm for self-adaptation in service-oriented systems," *ECOWS '09. Seventh IEEE European Conference on Web Services*, pp. 161 – 170, 2009.

[11] B. Pickering, S. Robert, S. Mnoret, and E. Mengusoglu, "Model-driven management of complex systems," *MoDELS'08. Proceedings of the 3rd International Workshop on Models@Runtime , Toulouse, France*, pp. 277 – 286, October 2008.

[12] V. Cardellini, E. Casalicchio, V. Grassi, F. L. Presti, and R. Mirandola, "Qos-driven runtime adaptation of service oriented architectures," *ESEC/FSE '09. 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.

[13] R. Nzekwa, R. Rouvoy, and L. Seinturier, "Modelling feedback control loops for self-adaptive systems," *Third International DisCoTec Workshop on Context-Aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (2010) 7*, vol. 28, 2010.

[14] V. K. Naik, A. Mohindra, and D. F. Bantz, "An architecture for the coordination of system management services," *IBM SYSTEMS JOURNAL*, vol. 43, no. 1, pp. 78–96, 2004.

[15] R. Quitadamo and F. Zambonelli, "Autonomic communication services: a new challenge for software agents," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 457–475, 2008.

[16] B. A. Caprarescu and D. Petcu, "A self-organizing feedback loop for autonomic computing," *IEEE CS'09. In Proceedings of the Computation World: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns*, vol. 126–131, 2009.

[17] R. J. Anthony, "Emergence: a paradigm for robust and scalable distributed applications," *ICAC'04. IEEE International Conference on Autonomic Computing*, pp. 132–139, 2004.

[18] C. Dorn and S. Dustdar, "Interaction-driven self-adaptation of service ensembles," *CAiSE'10. 2nd International Conference on Advanced Information Systems Engineering*, 2010.

[19] W. Truszkowski, L. Hallock, C. Rouff, J. Karlin, J. Rash, M. G.Hinchey, and R. Sterritt, *Autonomous and Autonomic Systems*. Springer, 2009.

[20] T. B. Sheridan, *Telerobotics, Automation, and Human Supervisory Control*. The MIT Press, 1992.

[21] R. W. Proud, J. J. Hart, and R. B. Mrozinski, "Methods for determining the level of autonomy to design into a human spaceflight vehicle: A function specific approach," *PerMIS 03. Proc. Performance Metrics for Intelligent Systems, NIST Special Publication 1014*, September 2003.

[22] D. A. Norman, A. Ortony, and D. M. Russell, "Affect and machine design: Lessons for the development of autonomous machines," *IBM Systems Journal*, vol. 42, no. 1, pp. 38 – 44, 2003.

[23] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A concise introduction to autonomic computing," *Elsevier Journal on Advanced Engineering Informatics,*, pp. 181–187, 2005.

[24] IEEE Standards Board, "IEEE Standard Glossary of Software Engineering Terminology (IEEE Std. 610.121990)," Tech. Rep. 610.121990, IEEE, August 1990.

[25] R. Anthony, D. Chen, M. Pelc, M. Persson, and M. Torngren, "Context-aware adaptation in dyscas," *CAMPUS'09. Proceedings of the Second International DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services*, vol. 19, 2009.

[26] S. A. Jacklin, M. R. Lowry, J. M. Schumann, P. P. Gupta, J. T. Bosworth, E. Zavala, J. W. Kelly, K. J. Hayhurst, and C. M. Belcastro, "Verification, validation, and certification challenges for adaptive flight-critical control system software," *In American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation and Control Conference and Exhibit, number AIAA 2004-5258, Providence, Rhode Island*, August 2004.

[27] B. T. Clough, "Metrics, schmetrics! how the heck do you determine a uavs autonomy anyway?," *Proceedings of the Performance Metrics for Intelligent Systems Workshop, Gaithersburg, Maryland*, 2002.

[28] International Organization for Standardization/International Electrotechnical Commission, "Software engineering Product quality (ISO/IEC 9126)," tech. rep., International Organization for Standardization/International Electrotechnical Commission, 1998.

The Challenge of Validation for Autonomic and Self-Managing Systems

Thaddeus O. Eze, Richard J. Anthony, Chris Walshaw and Alan Soper
Autonomic Computing Research Group
School of Computing & Mathematical Sciences (CMS)
University of Greenwich
Park Row, SE10 9LS, London, United Kingdom
{T.O.Eze, R.J.Anthony, C.Walshaw and A.J.Soper}@gre.ac.uk

***Abstract* -** The research community has achieved great success in building autonomic systems (AS) in line with the vision of autonomic computing (AC) released by IBM in 2001. The success is gaining ground in addressing the perceived concerns of complexity and total cost of ownership of information technology (IT) systems. But we are now faced with a challenge springing from this very success. This challenge is *trustworthiness* and there are limited research results published in this direction. This, if not addressed will definitely undermine the success of AC. How do we validate a system to show that it is capable of achieving a desired result under expected range of contexts and environmental conditions and beyond? This paper identifies the challenges and significance of AS validation and proposes a roadmap towards achieving trustworthiness in autonomic systems.

Keywords-trustworthiness; autonomicity; validation; certification

I. INTRODUCTION

IBM in 2001 observed in a manifesto [1] that the main obstacle to the future growth of IT is a looming software complexity crisis. The innovations in software development have increased exponentially, the sophistication and complexity of system design thereby stretching human capabilities to the limits to install, configure, optimise, maintain and manage these systems. The software is so complex that almost no single person knows everything about the software anymore. Another issue is how upgrades are handled; it is not clear whether or not an upgrade in one part of a system will result in loss of functionalities in other parts that integrate with it.

AC is chosen as a way forward [2]; the idea of building self-managing computing systems in the same fashion as the biological autonomic nervous system using high-level policy objectives set by human administrators. Though complete AS do not yet exist, some products from leading AC enterprises now claim to have self-managing features [3]. We have also seen great level of research interest in AC. A large number of surveys e.g., [4][3][5] have considered the work along a number of criteria and dimensions. In [4] the paper looked at AC to highlight which characteristics are necessary to evaluate and compare AS and derive definitive metrics for the evaluation. The survey in [3] categorizes complexity in IT systems and identifies which AC self-* properties address which complexity. [5] looks at IBM's MAPE-K (monitor, analyze, plan, execute and knowledge) control loop and identifies works that have been done in each of its components and also proposes an alternative to IBM's method of measuring system autonomicity [6]. There is also progress in injecting autonomic capabilities into legacy systems [7]. Other efforts such as those reported in [8][9] focus on using policy-based autonomic techniques to build generic AC frameworks arguing that the integration of techniques gives rather greater flexibility and more powerful adaptation than the individual techniques. With this huge effort devoted to the design and development of ASs, emphasis is lacking on the certification of these systems. We suggest that ASs must reach trustworthy status and be 'certifiable' to achieve the full vision of AC. Appropriate measures for validating AS decision-making processes should be defined. We identify this as the core challenge facing the success of AC. This is our main research focus. Another major problem facing the AC research field is the lack of standards. We have seen proliferation of approaches and the misuse of AC terms –different terms mean different things to different researchers. This shortcoming can only be addressed by standards.

Certification of ASs is a specific work area that needs attention and we believe this can be achieved through defining proper AS validation mechanisms. AC systems are designed and deployed across many application domains to address the challenge of human management complexities. We may come to a point where these systems take over full control of operations in those domains (e.g., businesses, military, health etc.) and any failure can be extremely costly –in terms of down time, danger to life, loss of control etc. This underpins the criticality of AS validation. Robust self-management in AC systems resulting in dynamic changes and reconfigurations requires that ASs should be able to continuously perform self-validation of their own behaviour and configuration, against their high-level behavioural goals and be able to reflect on the quality of their own adaptation behaviour. Such systems are considered trustworthy and then certifiable. It is then necessary to have a testing approach that combines design/run-time elements and is also an integral part of the self-management architecture. We have a longer term vision to develop certifiable systems. By trustworthiness, we mean a state where we can be confident that an AS will remain correct in the face of any possible contexts and environmental inputs and sequences of these; this is achieved through robust validation.

Our motivation is driven by the identified challenge of lack of certifiable ASs. In this paper we present a roadmap towards achieving certifiable AC systems by defining a layered autonomic solution architecture that incorporates validation as an integral part of the self-management structure. In our proposal, we identify the features and define a proper validation approach as one that is generic, cuts across design/run-time, and is an integral part of the whole self-management structure. Currently, most AS are tested in the same way other software is tested: *unit testing*. This includes simulations for performance analysis. In [10] it is suggested that a complete testing plan will require developmental stage-by-stage testing. This approach is limited because it is only design-time based and cannot guarantee trustworthiness. Though autonomic solutions methods establish system policies at design-time, AS must be able to deal with unforeseen conditions (unpredicted at design-time) that might arise during run-time and with this comes the possibility of ASs to deviate from intended behaviour and/or yield inconsistent results. As a result, what is needed is a system of validation that will not only test AS behaviour at design-time but also tests the system's behaviour under environmental circumstances or contexts not predictable at design time. It is not the component's behavior that is of key focus in the sense of being unknown – it is the circumstances in which it executes which is fundamentally unknown – and this may in turn cause unknown behavior in the component we are testing – or indeed in the whole system as a result. Only few works e.g., [11][12] provide means of run-time testing of AS adapted behaviours by introducing self-testing activities to ASs. The main goal of this paper is to outline challenges in current AS validation methods and propose a strategy leading to the achievement of certification of autonomic systems.

The remainder of this paper is organised as follows: Background, significance and challenges of AS validation are discussed in Section II. Analysis of identified validation techniques is presented in Section III. We propose a roadmap towards AS trustworthiness in Section IV and conclude the work in Section V.

II. BACKGROUND OF AS VALIDATION

In this section we define the problem of trustworthiness, identifying its significance and the extent of its challenge. We believe that the ultimate goal of AC should be the certification of AC systems. Yet to achieve certification requires a process and the meeting of some conditions (explained with Figure 1). For unknown reasons and in a bid to get things working faster, the AC research community has concentrated efforts on designs and architecture with little or no emphasis on system validation. Only very few researchers have identified trustworthiness as a major AC challenge and yet fewer [11][12][21] have actually suggested or proposed techniques. The problem in clear terms is the ignoring of AS trustworthiness and the general lack of validation efforts that specifically target the dynamic aspects of these systems.

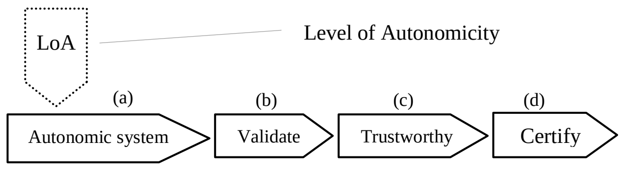


Figure 1: Proposed Certification process and requirements

Figure 1 represents a section in the journey towards full AC. At point (a) we assume that a system is developed and is considered autonomous at some level. This level is determined by a LoA measurement methodology which needs some form of standardization. The definition of LoA at this point is prerequisite to the next step. At point (b) is the system's self-validation distributed across design-time and run-time. When it is ascertained that a system is validated then it is trustworthy and trustworthiness is a vital foundational step on the road towards certification. It then follows that for a system to be certified, it must be trusted and only validated systems can be trusted. We draw a conclusion here that *for an AS to be certifiable there must be a standard for measuring the level and extent of its autonomicity* as it makes no meaning to certify a system whose extent of autonomic capability cannot be measured.

A. *Significance of The Problem*

The consequence of a lack of validation comes in two dimensions. On the one hand is the risk of losing control and loss of confidence that the autonomic system will not fail. This is obvious owing to the nature of ASs which includes dynamic changes caused by the self-* features in unpredictable environments and conditions. On the other hand is the issue of standardization of AS design processes. It is very unlikely to secure standards for invalidated systems as the general standardisation of a system will largely depend on the level of confirmation of its 'process correctness'. Process correctness is the demonstration of the correctness of a system's behaviour under a range of environmental conditions. According to [13] the requirement for determining process correctness arises from the human fear of selfish and uncontrolled behaviours that potentially might emerge inside self-managed systems. This is yet another factor that underpins the urgent need for standardisation in autonomics. We believe that once there is validation in place to ensure process correctness, standardization will be achieveable. Despite the huge effort in AC there are no known standards in the field [9]. The lack of certification and standards leads to proliferation of designs.

B. *Extent of The Challenge of Trustworthiness*

Trustworthiness is a broad area that needs extensive investigation and requires a carefully thought-out approach. In this section we look at the challenges of achieving AS trustworthiness and what form a trust solution must take.

- **Level of Autonomicy (LoA):** We have identified in Figure 1 the role of defining LoA in AS certification. LoA is a way of categorising ASs according to degrees; levels of dependences on humans for decision-making. Autonomic systems come in different degrees. Take for instance, one UAV (Unmanned Aerial Vehicle) that has a ground human pilot and one that hasn't are both autonomic systems but of different capacities. Now certifying both systems will demand different requirements as the later system will obviously need to meet higher requirements being more autonomic than the first. Classifying systems will give us an idea of their full capabilities and also identifies what conditions they must meet for certification. It also explains what level or extent of validation that is needed. This point is supported in [14] where it is argued that for a UAV to be certified, the level of its autonomicity should first be established to point out the direction and level of its certification. Currently, there is no agreed method of classifying AS and the only solution to this is standardization of approaches.

- **Design-time and run-time consideration:** As we have identified earlier, validation approaches will need to take care of both design-time and run-time owing to the nature of AC. Validating design-time decision making process does not suffice as systems' decisions that handle evolving conditions also need to be validated. The challenge here is extending validation algorithms to deal with run-time changes that may or may not be anticipated at design. For ordinary software applications where outcomes are as expected or predicted, design-time validation can suffice.

- **Reusability:** Validation approaches should be generic in nature –i.e. approaches should not be specifically defined for given self-adaptation processes. They should be adaptable to different processes. But with LoA in mind, approaches are expected to be generic within levels. What that means is that the reusability of approaches will be restricted according to LoA.

- **Robustness:** Validation solutions must show consistency with the dynamism of the AC environment. In other words solutions should also be autonomic in their approach. Validation approaches are also expected to be integral parts of the AC process.

III. CURRENT APPROACHES TO AS VALIDATION

In this section we survey a cross section of validation and testing approaches. These have been predominantly design-time or laboratory based although the nature of system in question determines, to a large extent, the type of validation or testing needed. Other forms of testing include simulations for performance analysis and self-testing approaches. We group the approaches as follows:

Unit Testing: Unit testing deals with the testing of known testable parts of a system. Usually the tested part, at the point of test, has definite (or well known) functions and outcomes. At this level of testing are laboratory and simulation based testing. [15] has proposed AML (Agent Modelling Language), a visual simulation software that models systems operations and concepts that are multi-agent based. Since AS are multi-agent based, this simulator makes good case for modelling the operations of individual agents (Autonomic Elements –AEs in this case). The paper shows how AML can be used to '*comprehensively*' and '*efficiently*' model the NASA's Prospecting Asteroids Mission (PAM) system. However, this simulation based validation does not suffice for AS trustworthiness as it depends on (or is limited to) the designer's knowledge of the system's environment and operations. Simon Dobson in [16] attempted adaptive network calculus which allows for both design and verification of adaptive systems. In this approach, the description of the adaptive behaviour and its verification are done mathematically; network calculus. This method however is specifically for network functionality. For example, in its expression it is assumed by definition that $\boldsymbol{R^*(t) \geq R(t)}$ for all $\boldsymbol{t}$. Where $\boldsymbol{R(t)}$ and $\boldsymbol{R^*(t)}$ define, respectively, the sum of bytes received and that of output by a network element at time $\boldsymbol{t}$. For us this expression can only hold under ideal circumstances (for systems of well known and predictable service curves) but cannot hold in a dramatically different set of circumstances beyond the design-time expected conditions.

Real Life: Testing and validation in a live system is arguably the most accurate way of verifying a system to ensure its compliance with the set system's goal. We understand that not all systems can be exposed to real life validation nonetheless researchers have identified it as one of the main approaches. In [17] the VisLab at the University of Parma, Italy, in seeking for new ways to test (validate) their developed autonomic vehicle for the 2010 World Expo in China, decided to drive their autonomic vehicles (through real-world traffic) 13000Km from Parma to China. Alberto Broggi (VisLab's director) says "*When you do things in the lab, it all really works. But when you go out in the real road, with real traffic, real weather, it's another story.*"

Pervasive Supervision: Pervasive supervision is a monitoring approach proposed in [18] to ensure process correctness of ASs. The supervision system is designed to continuously monitor the (known) configurations of each AE, interpret the monitored data according to certain operations requirements, like functional correctness, performance, consistency etc., and enforce corrective measures in case of requirement(s) violation. An example of pervasive self-supervision is found in [22] in which the policy mechanism monitors its own rate of decision change.

It was found that if a policy is faced with a situation where the utility of two outcomes are very close it is possible for the policy to rapidly switch between these options - effectively adding noise into the system for no additional benefit. By monitoring its own rate of decision change, the policy mechanism is able to detect this instability and temporarily shuts down its 'execute' function whilst continuing to run the 'monitor', 'plan' and 'analyse' components. A similar approach, *model checking*, is found in [19]. It provides automatic analysis of models for adherence to specified properties. A kind of logic is used to specify model properties that should hold during adaptation processes and these properties are automatically checked for adherence so as to provide assurance.

Self-testing: Self-testing is an approach to allow the managed system to carry out self validation of its decision making process. In [11] a framework to validate change requests in ASs is proposed. The approach is based on extending the current MAPE-based autonomic structure to include self-testing as an integral and implicit part of the AS. The same model or structure for AS management using autonomic managers (AMs) is replicated for the self-testing. In the self-test structure, test managers TMs (which extend the concept of AMs to testing activities) implement closed control loops on AMs (such as AMs implement on managed resources) to validate change requests generated by AMs. The work in [11] is extended in [20] to include auxiliary test services components that facilitate manual test management and a detailed description of interactions between the TMs and these new components. [21] proposes a reusable object-oriented design for developing self-testable autonomic software by providing a detailed reusable design for AMs, TMs, touchpoints and also extending the proposed self-testing framework in [11] to include knowledge sources to testing activities in autonomic software. This design (proposed for autonomic software systems) also applies the concepts of AMs. Arguing that these approaches are *not* generic, [12] has proposed a '*generic self-test approach*'.

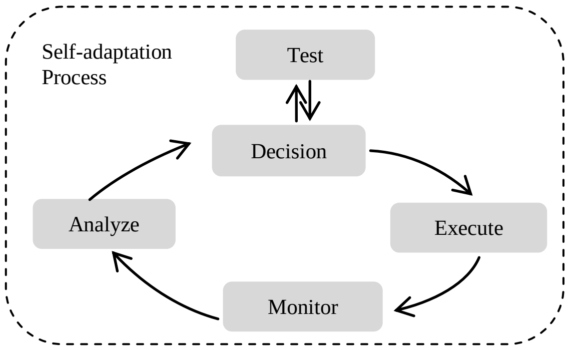


Figure 2: New control loop with test activity [12]

In the structure presented in Figure 2 the authors of [12] extended IBM's MAPE control loop to include a new function called *Test*. By this they define a new control loop comprising Monitor, Analyze, Decision, Test and Execute – MADTE activities. The MADTE loop works like the MAPE loop only that the *Decision* activity calls the *Test* activity to validate a chosen action should it determine to adapt a suggested behaviour. The *Test* activity carries out a test on the action and returns its result to the *Decision* activity which then decides whether to implement, skip or choose another action. (An adaptation is favoured if *Test* indicates that it will lead to component's better performance in terms of characteristics such as optimization, robustness or security.) The process is repeated if the latter is the case. When an action is decided on, the decision activity passes it to the *Execute* activity for implementation. A general method for testing context aware applications, which in a way simplifies the understanding of self-testing in AS is presented in [23]. The paper simplifies the concept of system management using context information while also testing the whole process including the interactions within.

A. *Taxonomy of Validation Approaches*

Our research has shown that different approaches can be used complementarily depending on what system is being tested and validated. In some cases, for example, using the software environment (simulation) to build a testing setup makes it relatively easier (and complementary) to build models for the real-world testing.

Table 1: Taxonomy of validation approaches

Validation Approach		Generic	Design-time	Run-time	Integrated
Unit Testing	Sim. [15]	√	√	−	−
	Lab. [8]	√	√	−	−
	[16]	√	√	−	−
Real World	[17]	√	√	√	NA
Pervasive Supervision / Model Checking	[18]	√	−	√	√
	[19]	−	−	√	−
Self-testing	[11]	−	−	√	√
	[12]	√	−	√	√
	[21]	√	−	√	√

We present the taxonomy of approaches under some selected properties as shown in Table 1. (Note that Approaches and referenced works are selected examples directly related to AS validation and so not exhaustive.). By Generic, we mean approaches that can be adapted to different adaptation processes. *Design-time* and *Run-time* indicate approaches that are design-time based and run-time based respectively. By *Integrated*, we mean approaches that are not separated from the autonomic management architecture. In some senses real world testing is more of testing than validation. It actually shows whether or not a particular AS is able to make appropriate decision(s) in the face of any change but says nothing about scrutinizing the decision(s) before implementing them. In the end, humans are needed to methodologically justify the decision(s) made. In our view, pervasive supervision is more of a component control paradigm than a validation method. In simple terms, pervasive supervision makes sure that a component takes a defined action (according to specified configurations) when a change occurs (e.g., contextual) but doesn't bother with validating the action taken. The works in [12][21] meet all the properties with the exception of design-time. The difference between [12] and [21] is that [21] defines a separate test loop (consistent with the self-management control loop) and integrates both loops while [12] integrates testing to the self-management control loop. Again the architecture in [21] is more complex than that in [12].

What we have discovered so far is that AS validation is much in its earliest stages with only self-testing as a promising approach amongst all identified approaches. What is then needed is a more robust and less complex self-testing validation methodology for AC systems. Research has also shown that AS validation is still much underdeveloped in some areas e.g., with respect to reusability of validation techniques. Though researchers claim their approaches are generic, it is not yet clear to what extent this is true considering the level of tweaking that needs to be done. In [12] for example, necessary actions to make the proposed framework generic are listed. This can be argued in terms of its robustness and the expected range of application domains (or self-adaptation processes) to be covered. It then follows that techniques for reusable validation is another research area needing attention.

IV. OUR PROPOSED ROADMAP TOWARDS AS TRUSTWORTHINESS

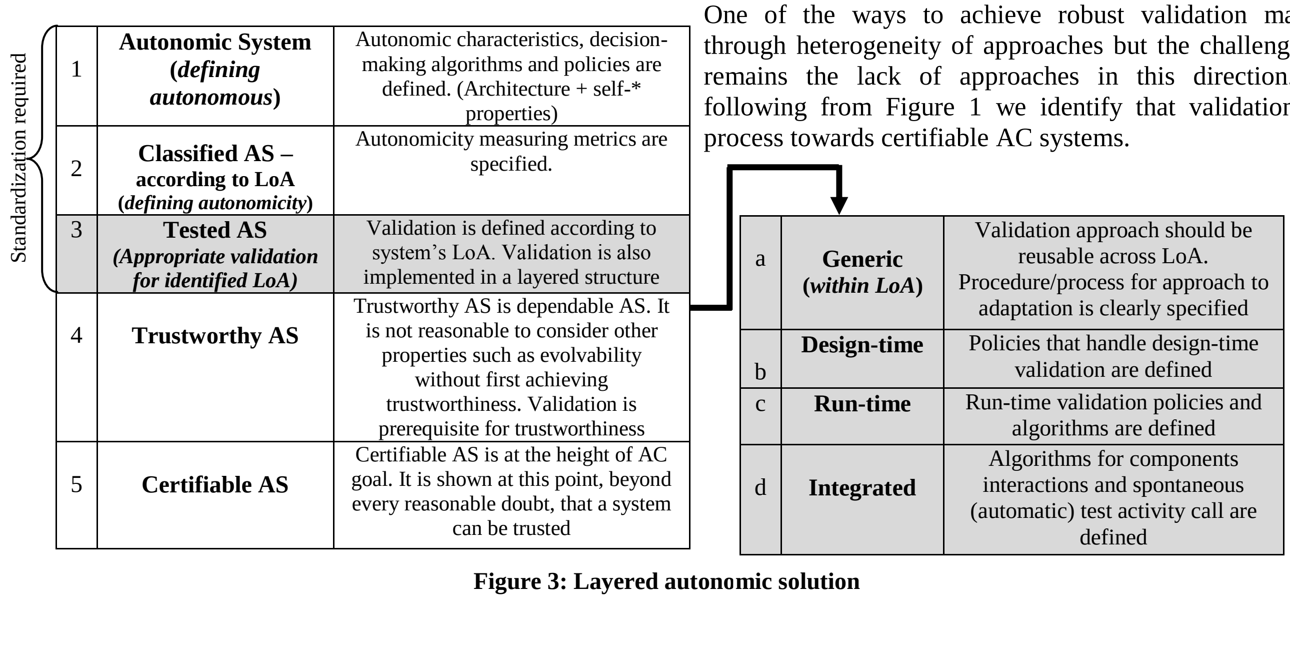
From the ongoing investigation discussed above, we now draw up a roadmap towards AS trustworthiness. This entails our view on how to achieve certifiable AC systems. This is a long road but it is vital that we take steps along this road. This forms the basis of our research strategy. First, we identify characteristics or features, if you like, a proper validation approach in our opinion should possess. Then we look at the inter-related steps towards certifiable Ass.

A. Features of AS Validation Approach

It is our opinion that a proper validation approach should have the characteristics shown in Table 1. ***Generic:*** Reusability reduces complexity and cost (in terms of time and effort) in developing validation processes for AS. A good validation approach should be flexible to be adapted to different adaptation processes and the procedure or process for this adaptation clearly detailed. ***Design/Run-time:*** The dynamic changes and reconfigurations in AS could result in drawbacks such as the possibilities of policy conflicts and incorrect goal specifications. Again it is clear that some AS frameworks facilitate decision-making both at design-time and run-time. It is then necessary to consider testing both at design-time and run-time. ***Integrated:*** In our view testing should be an integral part of the whole self-management architecture. Testing being integrated to the management structure achieves real time validation which is necessary to mitigate adaptation conflicts and promote consistency. ***Automatic***: We emphasize the importance of self-validation. Validation activity should be human independent (i.e. should be triggered by a change in application context, environmental volatility or a locally-detected failure requiring reconfiguration) following a defined validation process. But proving that a validation mechanism actually meets its set requirements is another issue of concern.

B. Towards Certifiable AC Systems

We define a proper validation approach as one that is generic, cuts across design/run-time, and is an integral part of the whole self-management structure. These features can be defined and implemented in a '*class'* architecture, i.e. each feature being seen as a class thereby defining four classes (a – d in Figure 3). This makes the design process flexible as it allows designers to tackle features within the boundaries of their separate classes. Take the *integrated* feature for example; at this class the designer will be concerned with such issues as defining algorithms for components interactions, spontaneous test activity call, etc. One of the ways to achieve robust validation may be through heterogeneity of approaches but the challenge still remains the lack of approaches in this direction. But following from Figure 1 we identify that validation is a process towards certifiable AC systems.



1	**Autonomic System (*defining autonomous*)**	Autonomic characteristics, decision-making algorithms and policies are defined. (Architecture + self-* properties)
2	**Classified AS – according to LoA (*defining autonomicity*)**	Autonomicity measuring metrics are specified.
3	**Tested AS *(Appropriate validation for identified LoA)***	Validation is defined according to system's LoA. Validation is also implemented in a layered structure
4	**Trustworthy AS**	Trustworthy AS is dependable AS. It is not reasonable to consider other properties such as evolvability without first achieving trustworthiness. Validation is prerequisite for trustworthiness
5	**Certifiable AS**	Certifiable AS is at the height of AC goal. It is shown at this point, beyond every reasonable doubt, that a system can be trusted

a	**Generic (*within LoA*)**	Validation approach should be reusable across LoA. Procedure/process for approach to adaptation is clearly specified
b	**Design-time**	Policies that handle design-time validation are defined
c	**Run-time**	Run-time validation policies and algorithms are defined
d	**Integrated**	Algorithms for components interactions and spontaneous (automatic) test activity call are defined

Figure 3: Layered autonomic solution

If we consider (1 - 5) in Figure 3 as layers, each layer is characterized by different attributes. To put it in more context we define three cardinal processes in the path towards certifiable AS; (1) *defining autonomous* – acceptable characteristics that define an autonomic system. This includes issues such as AC architecture (including decision-making algorithms) and self-* properties. The defined characteristics will influence the design and structure of the LoA in layer 2; (2) *defining autonomicity* (LoA) –this is concerned with the whole study of classifying AC systems according to the level of machine or human dependency. This entails measuring or calibrating AS using such metrics as LoA [24]. The designed AS is measured in (2) to determine its autonomic capacity. This gives an idea of the required validation for the system; (3) *validation of systems* –following the capacity of the system an appropriate validation mechanism is mapped out in layer 3 following the validation sub-layer (defined as classes *a* - *d*). These three processes are separate research areas and standards are required at each level. The roadmap identifies that a conceived AS is first designed following a predefined architecture, evaluated according to a set of autonomic characteristics to determine LoA and then validated against its goal. A trustworthy AS is achieved when the design of an AS follows layers 1 to 3. A trustworthy AS is dependable thereby making it certifiable.

V. CONCLUSION AND FUTURE WORK

Designing an autonomic and self-managing system is one thing while validating its management processes is another. Notwithstanding the emergence of products now claiming to have self-managing features, there is very little research effort towards the validation of AS and hence there are no known trustworthy AC systems. We have evaluated some proposed validation approaches against some features which are *generic, design-time, run-time, integrated*, and *automatic*. We have identified the importance of validation and measuring AS level of autonomicity to achieving certifiable AS, outlined challenges in current validation methods and have proposed a roadmap towards certifiable AC systems. As a future work we will be addressing the three cardinal processes identified in Figure 3 which includes autonomic solution architecture, methodology for measuring LoA and developing validation approaches.

VI. REFERENCES

[1] Horn Paul, *Autonomic computing: IBM perspective on the state of information technology*, IBM T.J. Watson Labs, NY, 15th October 2001. Presented at AGENDA 2001, Scottsdale.

[2] J. O. Kephart and D. M. Chess, *The vision of autonomic computing*, In IEEE Computer, volume 36, pp 41–50, January 2003

[3] Mazeiar Salehie and Ladan Tahvildari, *Autonomic Computing: Emerging Trends and Open Problems*, Workshop on the Design and Evolution of Autonomic Application Software (DEAS 2005), St. Louis, Missouri, USA, 2005

[4] J. A. McCann and M. C. Huebscher, *Evaluation issues in Autonomic Computing*, Grid and Corporative Computing (GCC) Workshop, LNCS 3252, pp. 597-608, Springer-V erlag, Birlin Heidelber, 2004

[5] Huebscher M. C. and McCann J. A., *A survey of autonomic computing—degrees, models, and applications*, ACM Computer Survey, 40, 3, Article 7 (August 2008)

[6] IBM Autonomic Computing White Paper, *An architectural blueprint for autonomic computing*. 3rd edition, June 2005, pp 25

[7] Kaiser G, Parekh J, Gross P, and Valetto G., *Kinesthetics extreme: an external infrastructure for monitoring distributed legacy systems*. In: Proceedings of the AC workshop, 5th international workshop on active middleware services (AMS), Seattle. pp 22–30, 2003

[8] Radu Calinescu, *General-Purpose Autonomic Computing*, In: Autonomic Computing and Networking, SpringerLink, 2009

[9] Richard John Anthony, *Policy-centric Integration and Dynamic Composition of Autonomic Computing Techniques*, Int'l Conference on Autonomic Computing (ICAC), June 2007, Jacksonville, FL

[10] Walt Truszkowski, Lou Hallock, Christopher Rouff, Jay Karlin, James Rash, Michael G. Hinchey and Roy Sterritt, Autonomous and Autonomic Syste*ms: with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*, Springer-Verlag London Ltd, London, 2009

[11] Tariq King, Djuradj Babich, Jonatan Alava, Peter Clarke and Ronald Stevens, *Towards Self-Testing in Autonomic Computing Systems*, Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07), Arizona, USA, 2007

[12] Andrew Diniz, Viviane Torres and Carlos José, *A Self-adaptive Process that Incorporates a Self-test Activity*, Monografias em Ciência da Computação, No. 32/09, Rio – Brasil, Nov. 2009.

[13] Mikhail Smirnov, Jens Tiemann, Ranganai Chaparadza, Yacine Rebahi and Symeon Papavassiliou, *Demystifying self-awareness of autonomic systems*, Conference Proceedings ICT-MobileSummit 2009, Santander, Spain. Dublin: IIMC, 2009, pp 9.

[14] R. D. Alexander, M. Hall-May and T. P. Kelly, *Certification of Autonomous Systems under UK Military Safety Standards*, ScientificCommons, 2007

[15] Radovan Cervenka, Dominic Greenwood, and Ivan Trencansky, The AML Approach to Modeling Autonomic Systems, International Conference on Autonomic and Autonomous Systems (ICAS'06), Silicon Valley, USA, July 19-21, 2006.

[16] Simon Dobson, *An adaptive systems perspective on network calculus, with applications to autonomic control*, Int. J. Autonomous and Adaptive Communications Systems, Vol. 1, No. 3, pp.332–341. 2008

[17] Erico Guizzo, *Autonomous Vehicle Driving From Italy to China*, IEEE Spectrum tech alert, 23rd Sept. 2010

[18] Luciano B., Matthias B., Maurice M., Chris N., Kevin C. and Peter H., *Towards Pervasive Supervision for Autonomic Systems*, Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence & Its Applications, 2006

[19] J. Zhang, H. J. Goldsby, and Betty H.C. Cheng. *Modular verification of dynamically adaptive systems*. In Proceedings of the 8th Int'l Conference on Aspect-Oriented Software Development, 2009.

[20] Tariq M. King, Alain E. Ramirez, Rodolfo Cruz, and Peter J. Clarke, *An Integrated Self-Testing Framework for Autonomic Computing Systems*, Journal of computers, vol. 2, no. 9, november 2007

[21] Tariq M. King, Alain Ramirez, Peter J. Clarke, Barbara QuinonesMorales, *A Reusable ObjectOriented Design to Support SelfTestable Autonomic Software*, Proceedings of the 2008 ACM symposium on Applied computing, Fortaleza, Ceara, Brazil, 2008

[22] R. J. Anthony, *Policy-based autonomic computing with integral support for self-stabilisation*, International Journal of Autonomic Computing, Inderscience, Vol. 1, No. 1, pp.1-33. 2009.

[23] Stefan Taranu and Jens Tiemann, *General Method for Testing Context Aware Applications*, Proceedings of the 6th international workshop on Managing ubiquitous communications and services (MUCS), Barcelona, Spain, June 15, 2009

[24] Haffiz Shuaib, Richard J. Anthony and Mariusz Pelc, *Towards Certifiable Autonomic Computing Systems*, Technical report 1, Autonomic Research Group, CMS, University of Greenwich, 2010

Nature-inspired Self-organization in P2P Ad hoc Grids

Tariq Abdullah, Koen Bertels,
CE Laboratory, EEMCS, Faculty, Delft University of Technology,
Mekelweg 4, 2624 CD, Delft, The Netherlands
{m.t.abdullah, k.l.m.bertels}@tudelft.nl

***Abstract*—Resource availability fluctuates over time in ad hoc grids and other similar distributed systems due to their dynamism and complexity. These changes require adaptation of the system to the new system state. This paper will present an ant colony based self-managing mechanism in a P2P ad hoc grid. In this paper, we apply ant colony optimization for self-management of a P2P ad hoc grid. The proposed mechanism will enable the ad hoc grid participants to self-organize themselves and form/de-form virtual resource clusters according to their changing resource availability/requirements. We also investigate how can ants form a dynamic cluster/ring of specialized resources. The experimental results show that the proposed mechanism work better than the previously proposed mechanisms.**

Keywords– *Self-organization; natural intelligence; ad hoc grids*

I. INTRODUCTION

There are a large number of naturally existing Complex Adaptive Systems (CAS) like fish swim in schools, geese fly in organized V-shaped flocks, immune system, sand dune ripples and ant foraging. These CAS are dynamic, highly decentralized networks and consist of many participating agents and are characterized by decentralized control, emergent behavior, robustness and self-organization. The participating agents in these systems interact according to simple local rules which result in self-organization and complex behavior.

Ad hoc grids and similar computational distributed systems are inherently dynamic and complex systems. Resource availability fluctuates over time in ad hoc grids. These changes require adoption of the system to a new system state by applying some self-organizing mechanism.

We aim to study the different adaptation mechanisms that can be used to introduce self-organization in ad hoc grids (or similar distributed systems) in the context of GRAPPA project [1]. We envision Ant Colony Optimization (ACO) [2] as one way of introducing self-organization in distributed systems under varying resource availability of the autonomous participating nodes of the distributed systems. We are also interested in understanding the effect of ACO based mechanism on the infrastructural spectrum.

We proposed a dynamic, self-organizing mechanism for resource discovery in ad hoc grids in our previous work [3], where, we focused on developing dynamic and scalable mechanisms for resource discovery that could self-organize according to the work load of the resource manager (referred to as *matchmaker* hereafter).

In this paper, we propose an ant colony inspired mechanism for self-organization in ad hoc grid. The proposed self-organizing mechanism also focuses on the dynamic formation of resource cluster(s) according to resource demands. We apply the meta-heuristic methods from ACO for dynamic cluster formation/de-formation and resource virtualization in an ad hoc grid. In these meta-heuristic methods a complex, adaptive behavior emerges from the local interactions of the participating ants. Each ant (a participating node of the ad hoc grid) is looking for its required resources without any centralized control.

The rest of the paper is organized as follows. Related work is presented in Section II. Section III summarizes the micro-economic based modified ACO algorithm and formulas for pheromone calculation presented in our previous work [4]. Proposed nature inspired self-management mechanism in explained in Section IV. Experimental setup & experimental results discussion are in Section V. Whereas, Section VI concludes the paper with some future research directions.

II. RELATED WORK

Jaeger et al. [5] applied bloom filter for self-organizing broker topologies in publish/subscribe systems. Their work is closely related to the work presented in this paper. The main problem with their work is they only considered the similarity of the notification messages in publish/subscribe system to reduce the total cost of forwarding and processing the notification messages. Ritchie et al. [6] proposed a hybrid ant colony optimization algorithm to select the appropriate scheduler in a heterogeneous computing environment. The proposed approach was only tested in solving a scheduling problem in a static environment for independent jobs. Deng et al. [7] proposed a resource discovery mechanism for Peer-to-Peer (P2P) grids inspired by ant colony optimization algorithm. Fidanova et al. [8] attempted searching for the best task scheduling for grid computing using ant colony optimization based algorithm. Zeng et al. [9] proposed a dynamic load balancing mechanism based the count of waiting jobs and the arrival rate of the new jobs in distributed systems. Andrzejak et al. [10] compared different algorithms, including ant colony optimization algorithms, for self-organization and adaptive service placement in dynamic distributed environments. Messor [11] is implemented on top of the Anthill framework [12]. An ant can be in Search-Max or Search-Min states. The ant wander randomly in the environment it find an overloaded node. The same ant, then, changes its state to Search-Min and wanders randomly again in the environment, while looking for an underloaded node. After these state changes, the ant balances the underloaded and overloaded node. However, considering the dynamism of grid environments, this information may cause erroneous load balancing decision making.

The main contributions of this paper are that this paper proposes an ant colony inspired, micro-economic based, self-organizing mechanism in a P2P ad hoc grid. The proposed mechanism introduces self-organization in the ad hoc grid and enables it to re-organize and adapt to a stable status under varying network and work load conditions. The scalability and robustness of the proposed mechanism is tested on PlanetLab [13].

III. Micro-economic based ACO Algorithm

We presented and explained an ant colony inspired micro-economic based resource management mechanism for ad hoc grids in [4]. The algorithms, formulas and the mapping of an ACS to an ad hoc grid were explained in the perspective of a micro-economic domain. We provide an overview of the mechanism in this section for better understanding the results presented in this paper.

An ad hoc grid node acts like a consumer/producer node in the modified ACO algorithm, and the matchmaker(s) are treated like the *food sources*. Each consumer/producer node is capable of generating and sending *request/offer ants* to the food source. The *pheromone value* indicates the weight of the matchmaker in the ant system. A matchmaker with higher pheromone value indicates that it has a higher probability of finding a compatible resource offer for a submitted resource request and vice verse.

Each joining node in our ad hoc grid is under the responsibility of a matchmaker and sends its resource request/offer ants to its responsible matchmaker. The joining node gets the pheromone value of the food source from its responsible matchmaker. The pheromone value of a matchmaker is updated for each resource request or resource offer received from a consumer/producer node. A matchmaker exchanges periodically its pheromone value with its immediate neighboring matchmakers (the successor and the predecessor matchmakers). The updated pheromone value is then sent to the consumer/producer nodes with a matched message. The consumer/producer node uses the pheromone value as an indicator of matchmaker's matchmaking performance. The consumer/producer node sends its next request/offer ant to a food source with the highest pheromone value. The pheromone value of a matchmaker is calculated periodically according to the following formula [4] given below:

$$\tau_{new} = \begin{cases} \alpha * \tau_{old} + (1-\alpha) * \triangle\tau & if\ \triangle\tau > 0 \\ (1-\alpha) * \tau_{old} + \alpha * \triangle\tau & if\ \triangle\tau < 0 \end{cases} \quad (1)$$

The parameter α represents the pheromone evaporation rate. The value of α varies between 0 and 1. τ_{old} represents the pheromone value during time interval $T1 = [t_{s_1}, t_{e_1}]$. Whereas, $\triangle\tau$ is the change in the pheromone value between the time interval $T1 = [t_{s_1}, t_{e_1}]$ & $T2 = [t_{s_2}, t_{e_2}]$. The start time of both intervals is represented by t_{s_1} & t_{s_2} and t_{e_1} & t_{e_2} represent the end time of both the intervals, such that $T2 > T1$ & $t_{s_2} = t_{e_1}$. Value of $\triangle\tau$ is calculated as:

$$\triangle\tau = \sum_{i=1}^{n} \tau(i)/_N \quad (2)$$

where N is the total number of messages received by the matchmaker and $\tau(i)$ is the pheromone value contributed by an individual ant. $\tau(i)$ for a consumer agent is calculated as:

$$\tau(i) = Perform(MM) * UPrice_{consumer} \quad (3)$$

$\tau(i)$ for a producer agent is calculated as:

$$\tau(i) = Perform(MM) * UPrice_{producer} \quad (4)$$

where $Perform(MM)$ represents the performance of a matchmaker and $UPrice$ represents the unit price of a requested or offered computational resource by an ant. The above formulas are explained in [4].

IV. ACO based Self-managing Ad Hoc Grid Segmentation/De-segmentation

This section explains the proposed ACO based self-managed segmentation and de-segmentation approach in an ad hoc grid. This self-managed, dynamic segmentation and de-segmentation process is based on the dynamic changing resource requirements and resource availability in an ad hoc grid.

In the proposed mechanism, there can be as many segments of the ad hoc grid as many needed. These segments are created when needed and are removed, when not needed. The proposed approach also considers the ad hoc grid segmentation for specialized resource requirements of the participating ad hoc grid nodes. These resources may include a pool of specialized hardware for a video rendering application or a pool of software resources for a data processing application or a pool of resources for remote collaboration on a scientific experiment.

A consumer/producer node knows about its resource category at the time of joining the ad hoc grid. The consumer/producer node joins the ad hoc grid by contacting one of the existing nodes in the ad hoc grid, referred to as a bootstrap node. A consumer/producer node discovers a responsible matchmaker for its resource category. The algorithms for node joining and finding a responsible matchmaker are detailed in [14]. All of the newly joining consumer/producer nodes discover their responsible matchmakers and the nodes of a similar resource category are under the responsibility of one matchmaker. An ad hoc grid node can change its resource category or its resource availability/demand status (being a consumer or producer of resources) at any time during its life span. A consumer/producer node generates and sends a request/offer ant after finding its responsible matchmaker.

Algorithm 1 Discovering the right food source.

```
1:IF (($Ant_{resCategory}$ == $FS_{resCategory}$) AND ($FS_{phValue}$ is highest)) THEN
2:   CALL Find matching request/offer algorithm
3:    IF (no match found) THEN
4:       IF ($Ant_{resCategory}$ == $pFS/sFS_{resCategory}$ ) THEN
5:              $Ant$ visit $pFS/sFS$
6:     GO TO Step 1
7:          END IF
8:     END IF
9:     ELSE IF ($Ant_{resCategory}$ != $FS_{resCategory}$) THEN
10:    $Ant$ visit $successorFS$
11:     GO TO Step 1
12:     END IF
13:END IF
```

Algorithm 2 Find matching request/offer.

```
1:Store request/offer in request/offer repositories
2:Match request parameters with offer parameters
3:IF ($Offer_{parameters}$ match $Request_{parameters}$) THEN
4:      Send matched message to consumer
5:      Send matched message to producer
6:      Update pheromone value
7:END IF
```

The proposed mechanism comprises the algorithms for *discovering the right food source (matchmaker)*, *finding the matching request/offer* and *changing the segment*. A food source determines whether it is the right food source to process the received resource request/offer ants by executing the algorithm discovering the right food source (Section IV-A). The *right food source* processes the received request/offer ant and attempts finding a matching offer/request ant by executing the algorithm find matching request/offer (Section IV-B). The participating consumer/producer nodes are autonomous and can change their resource category at any time. When they change their resource category then they join the new resource segment. The steps for changing the resource segment of a consumer/producer node are performed by executing the algorithm change segment (Section IV-C).

A. Discovering the Right Food Source

A matchmaker receives a request/offer ant, and first checks whether it is the right food source for processing the received request/offer ant or not. This process is listed in Algorithm 1 and is performed as follows:

Before explaining Algorithm 1, we first introduce the notations used in this algorithm. The resource category of the received request/offer ant is denoted by $Ant_{resCategory}$, resource category of the current matchmaker is denoted by ($FS_{resCategory}$). Whereas, $pFS_{resCategory}$ and $pFS_{resCategory}$ denote the resource category of the successor and predecessor matchmaker resource category respectively. The pheromone value of the current matchmaker is denoted by $FS_{phValue}$.

After receiving a request/offer ant, the matchmaker first checks whether the resource category of the received request/offer ant ($Ant_{resCategory}$) is similar to its resource category ($FS_{resCategory}$). Secondly, the matchmaker also checks whether the predecessor's resource category ($pFS_{resCategory}$) and/or the successor's resource category ($sFS_{resCategory}$) is similar to its resource category. In case of a matched resource category, the matchmaker also checks that it has the highest pheromone value ($FS_{phValue}$).

In case the matchmaker does not have the matching offer/request then the request/offer ant is forwarded to the predecessor/successor matchmaker with the second highest pheromone value. The predecessor/successor matchmaker node processes the received request/offer ant in the same way as explained above. When the resource category of the request/offer ant ($Ant_{resCategory}$) is different from the resource category of the matchmaker ($FS_{resCategory}$) and its immediate neighboring matchmaker nodes (successor and predecessor matchmakers), then the request/offer ant is forwarded to the successor matchmaker node. The successor matchmaker node again performs the steps in Algorithm 1.

B. Find Matching Request/Offer

After determining from its local knowledge that it is the right matchmaker/food source to process the received request/offer ant, the matchmaker attempts finding a matching offer/request for the received request/offer ant by following the steps listed in Algorithm 2. The matchmaker first stores the received request/offer ants in it request/offer repositories. Then the matchmaker/ food source compares the offer parameters ($Offer_{parameters}$) with the constraints/ parameters of received request ($Request_{parameters}$). If a resource offer satisfies the resource request parameters then a matching resource request/offer pair is declared by the matchmaker and the matching consumer/producer nodes are directly notified by the matchmaker about the match[3].

After finding a successful match, the pheromone strength of that matchmaker increases. The pheromone value of a matchmaker is based on its matchmaking efficiency. The pheromone value of a matchmaker decreases, when it is unable to find a match. The matchmaker periodically notifies its immediate neighboring matchmaker nodes (successor and predecessor matchmaker nodes) about the change in its pheromone value. The consumer/producer nodes communicate directly with each other for task execution after receiving a matched request/offer message from the matchmaker. The producer node returns the results back to the consumer node after executing the task.

C. Change Segment

The process of changing a node's segment is triggered when a node changes its resource category. This process is listed in Algorithm 3 and is performed as follows: Whenever a consumer/producer node changes its resource category, it receives a matched message reply from a new matchmaker ($matchMsgSender_{FSID}$). A consumer/producer node

Algorithm 3 Change segment.

```
1:IF ($CPNode_{FSID}$ $!=$ $matchMsgSender_{FSID}$) THEN
2:      $CPNode_{FSID} == matchMsgSender_{FSID}$
3:      Join new virtual segment
4:      Send request/offer ant to $CPNode_{FSID}$
5:ELSE
6:      No change in virtual segment
7:      Send request/offer to old $CPNode_{FSID}$
8:END IF
```

changes its matchmaker node ($CPNode_{FSID}$) and sends its next request/offer ant to the new matchmaker. In this way, a consumer/producer node continues sending its subsequent request/offer ants to the matchmaker that sent a matched message reply. Whenever, an ad hoc grid node changes its resource category, it leaves its current virtual segment and becomes a member of another virtual segment. Therefore, the ad hoc grid keeps on changing its infrastructure and is divided into specialized resource segments. The consumer/producer nodes are not bound to a specific ad hoc grid segment. The nodes can dynamically leave one virtual segment and join another according to their changed resource category.

V. Experimental Results and Discussion

Both the experimental setup used for evaluating the proposed mechanism and the experimental results are discussed in this section.

A. Experimental Setup

The modified ACO algorithm is implemented by extending Pastry [15], a structured overlay network. Pastry forms an overlay network among the ad hoc grid nodes and performs the basic tasks required for maintaining an overlay network. The experimental results reported here are obtained by executing the experiments on PlanetLab [13].

These experiments are executed in a different network condition including balanced network (BN) condition, resource intensive network (RIN) condition and task intensive network (TIN) condition. The consumer-producer ratio is approximately $50 - 50$ in BN condition. Whereas, the consumer-producer ratio is $20 - 80$ and $80 - 20$ in RIN and TIN conditions, respectively. The number of participating nodes is varied from $15 - 650$. The number of different resource categories is 3 in the first set of experiments and number of matchmakers is 5 in the second set of experiments. Different parameters of resource request/offer like task execution time and resource quantity are randomly generated from a pre-specified range. The validity period (TTL) of request/offer message is set to 10000 milliseconds for accommodating delays observed in PlanetLab. The results of the BN condition are discussed in detail.

Each ant's pheromone is initialized by 1. In nature, each ant wanders randomly without any initial pheromone value as used in these experiments. However, if the initial pheromone value is set to 0, then the new pheromone value will always be zero.

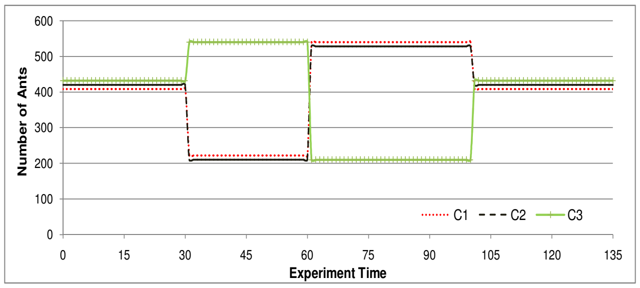


Figure 1: Workload distribution of different resource categories.

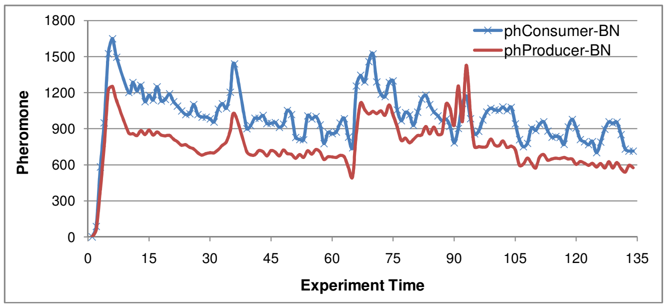
The value of α (rate of pheromone evaporation) is set to 0.8 in these experiments. The workload distribution among different resource categories is shown in Figure 1. In this figure, the experiment time is represented on the horizontal axis, and the number of ants of each resource category is shown on the vertical axis.

The analysis parameters are pheromone evaluation, consumer/producer utilization, response time, and the average ask/bid price of the participating producer/consumer nodes. The formulas for calculating the pheromone value are described in Section III. The consumer utilization and producer utilization are calculated according to the following equation: $(MatchedMessage/N)*100$, where $MatchedMessage$ represents the count of matched messages and N denotes the total number of request/offer ants processed by the matchmaker(s) in a unit time interval. The *response time* represents the time interval between receiving a request/offer message and finding a matching offer/request by the matchmaker. The response time is calculated as:$RT = T_{match} - T_{receive}$, where RT represents the response time, T_{match} is the time when the matchmaker found a matching offer/request for the received request/offer and $T_{receive}$ is the receiving time of the received request/offer. Following simplifying assumptions are in place for the experimental results reported in this paper and will be relaxed in the future work: (1) There exists at least one food source for a resource category. (2) Each consumer/producer node knows about or is under the responsibility of one food source at any given time. (3) The successor/predecessor food sources of a food source node (aka matchmaker node) update the current matchmaker node after updating their pheromone value and vice verse.

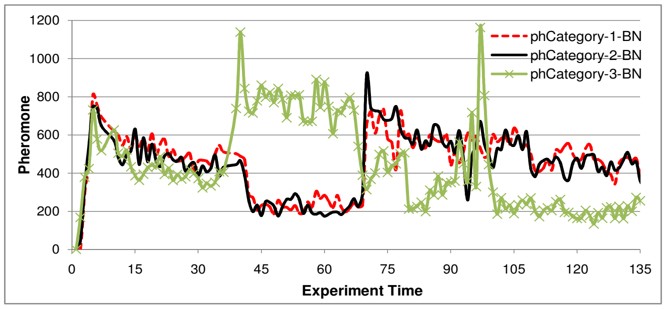
B. Experimental Results Discussion

The proposed ACO-based self-organizing mechanism is compared with different matchmaking schemes. These schemes include the simplest and less compute intensive first come, first served and a micro-economic based, compute intensive continuous double auction scheme.

The overall pheromone evolution for each resource category (represented as $phCategory-1-BN$, $phCategory-2-BN$ and $phCategory-3-BN$) during the simulation in a BN condition is depicted in Figure 2b. The experiment time is represented on the horizontal axis and the pheromone value

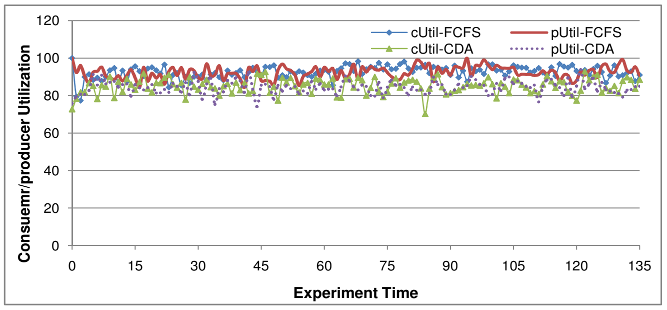


(a)

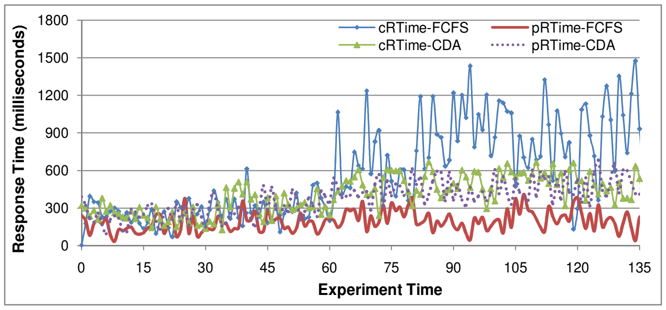


(b)

Figure 2: **(a)** Consumer/producer pheromone evolution in BN condition of the ad hoc grid. **(b)** Individual category pheromone evolution in an ad hoc grid.



(a)



(b)

Figure 3: **(a)** Consumer/producer utilization **(b)** Consumer/producer response time of the in the ad hoc grid in BN condition.

is on the vertical axis. The pheromone of each category evolves according to the workload distribution of the respective resource category (Figure 1). Whenever the distribution of ants in different resource categories is changed, a change in the respective pheromone concentration is observed. The ad hoc grid self-organizes itself after each change. A stable pheromone value is observed before the next disturbance in the ant's distribution during the simulation.

The pheromone behavior of the individual consumer/producer nodes in BN condition in represented in Figure 2a. The horizontal axis of both these figures represent the experiment time and the vertical axis represents the pheromone value of the participating consumer/producer nodes of the ad hoc grid. The initial increase of the pheromone value for the consumer/producer nodes is followed by a decreasing trend that leads to a stable status of the ad hoc grid. Similar to the individual category pheromone value evolution, the overall consumer/producer pheromone also evolves similarly. The pheromone pattern is disturbed after a change of the ant's distribution in different resource categories. The proposed algorithm enables the ad hoc grid to re-structure itself into different virtual resource and in attaining a stable state.

The average consumer/producer utilization of the participating consumer/producer nodes in an ad hoc grid with different matchmaking mechanisms is depicted in Figure 3a.

In-spite of changing workload of ants in different resource categories, the consumer utilization ($cUtil-CDA$, $cUtil-FCFS$) and the producer utilization ($pUtil-CDA$, $pUtil-FCFS$) in CDA and FCFS schemes under a BN condition remains above 80%. The fluctuations in the consumer/producer utilization refer to the activity in the ad hoc grid. This behavior implies that the compute intensive nature of CDA does not affect the matchmaking capacity of the ad hoc grid. It can be concluded from the above discussion that, in spite of being compute intensive, the consumer/producer utilization in CDA is as good as in FCFS. Whereas, the effect of an overload condition on a matchmaker in the proposed mechanism and a solution for avoiding the overload condition was discussed in [4].

The proposed self-organizing mechanism can also be analyzed from the matchmaker response time for the consumer/producer nodes. The consumer response time and the producer response time for continuous double auction scheme and first come, first served scheme in a balanced network condition is represented in Figure 3b. The response time shows a stable behavior and is not affected by the resource category change of the participating consumer/producer nodes. The consumer response time ($cRTime-CDA$, $cRTime-FCFS$) and the producer response time ($pRTime-CDA$, $pRTime-FCFS$) for both the schemes in BN condition are initially low due to the equal number of ants of different categories. When the consumer/producer nodes change their resource category, the ad hoc grid becomes unstable. The consumer/producer nodes have to wait longer for getting a matched response. This longer wait time results in an increasing trend of the consumer/producer response time. The response time shows a stable behavior once the ad hoc grid attains back the stable behavior.

The consumer response time for FCFS ($cRTime-FCFS$) is higher than that of the consumer response time for CDA ($cRTime - CDA$) in a BN condition. The lower consumer response time in CDA can be understood by understanding the requests/offers handling process in the matchmaker request/offer repositories. The matchmaker stores requests in descending order and offers in ascending order of the price in its request/offer repositories. The consumer response time is less in CDA scheme as compared to the consumer response time in FCFS scheme, due to the sorted placement of requests/offers in the matchmaker repositories.

It can be concluded from the above discussion that a nature inspired ACO-based, self-organizing mechanism with CDA scheme is preferred over an ACO-based mechanism with FCFS scheme. CDA based mechanism performs as good as the FCFS mechanism in terms of consumer/producer utilization, response time and pheromone value. CDA based mechanism enables the individual consumer/producer nodes to value their resource requests and resource offers according to their previous experiences from the ad hoc grid. The consumer/producer nodes can increase/decrease their bid/ask prices according to the resource demand/availability in the ad hoc gird. Thus, a CDA based ACO mechanism enables the node level self-organization along with the system level self-organization.

VI. Concluding Remarks

An ACO inspired, micro-economic based resource management approach for the ad hoc grids is presented in this paper. We used matchmaking performance as the basic factor for calculating the pheromone value. The proposed ACO inspired CDA based approach enables node level as well as system level self-organization and supports resource specialization in an ad hoc grid. From the experimental results it can be concluded that the proposed mechanism gives a stable behavior of the system in resource management, and shows better load balancing.

References

[1] http://ce.et.tudelft.nl/grappa/, GRAPPA Project, last visited March-2011.

[2] M. Dorigo and C. Blum, “Ant colony optimization theory: A survey,” *Theoretical Computer Science*, vol. 344, pp. 243–278, 2005.

[3] T. Abdullah, V. Sokolov, B. Pourebrahimi, and K. Bertels, “Self-organizing dynamic ad hoc grids,” in *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, Venice, October 2008, pp. 202–207.

[4] T. Abdullah, K. Bertels, and L. O. Alima, “Ant colony inspired microeconomic based resource management in ad hoc grids,” in *Proceedings of the 4th International Conference on Grid and Pervasive Computing*, Geneva, May 2009, pp. 189–198.

[5] M. A. Jaeger, H. Parzyjegla, G. Muhl, and K. Herrmann, “Self-organizing broker topologies for publish/subscribe systems,” in *Proceedings of the ACM symposium on Applied computing (SAC’07)*, 2007, pp. 543–550.

[6] G. Ritchie and J. Levine, “A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments,” in *Proceedings of the 23rd workshop of the UK planning and scheduling special interest group*, Cork, Ireland, 2004, pp. 1–7.

[7] Y. Deng, F. Weng, and A. Ciura, “Ant colony optimization inspired resource discovery in P2P Grid systems,” *The Journal of Supercomputing*, vol. 49, no. 1, pp. 4–21, 2008.

[8] S. Fidanova and M. Durchova, “Ant algorithm for grid scheduling problem,” in *Proceedings of the 5th International Conference on Large-Scale Scientific Computations (LSCC ’05)*, 2006, pp. 405–412.

[9] Z. Zeng and B. Veeravalli, “Design and performance evaluation of queue-and-rate-adjustment dynamic load balancing policies for distributed networks,” *IEEE Transactions on Computer*, vol. 55, no. 11, pp. 1410–1422, 2006.

[10] A. Andrzejak, S. Graupner, V. Kotov, and H. Trinks, “Algorithms for self-organization and adaptive service placement in dynamic distributed systems,” HP laboratories Palo Alto, Tech. Rep. HPL-2002-259, 2002.

[11] A. Montresor, H. Meling, and O. Babaoglu, “Messor: Load-balancing through a swarm of autonomous agents,” University of Bologna, Tech. Rep., May 2002.

[12] Özalp Babaoglu, H. Meling, and A. Montresor, “Anthill: A framework for the development of agent-based peer-to-peer systems,” in *Proceedings of the 22nd IEEE International Conference on Distributed Computing Systems (ICDCS ’02)*, 2002, pp. 15–22.

[13] https://www.planet-lab.org/, PlanetLab Online, last visited March-2011.

[14] T. Abdullah, L. O. Alima, V. Sokolov, D. Calomme, and K. Bertels, “Hybrid resource discovery mechanism in ad hoc grid using strucutred overlay,” in *Proceedings of the 22nd International Conference on Architecture of Computing Systems*, March 2009, pp. 108–119.

[15] A. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in *Proceedings of the 18th IFIP/ACM International Conference on Distributed Systems Platforms (Middleware ’01)*, vol. 2218, 2001, pp. 329–350.

Aspects of Innateness and Introspection in Artificial Agents

Chris White, David Bell, Weiru Liu
Queens University Belfast,
School of Computer Science,
Belfast, UK
{cwhite06, da.bell, w.liu} @qub.ac.uk

Abstract - **Autonomous Computation and cognition, the set of processes that characterise intelligent behaviour are related. For example, cognition drives the autonomy of a robotic agent. In this paper, it is our position that Innateness and Introspection, sometimes referred to here as Instinct and Observation, respectively, are key areas not explicitly focused upon in many current cognitive architectures. By identifying a 'minimalistic' cognition, and incrementally adding Innateness capabilities and Introspection ability, this research defines a structure underpinning a robot explorer that can deal with uncertain environments. The flexibility offered by this structure is considered to be essential for full autonomy. This paper proposes a framework for achieving Innateness and Introspection in autonomous agents and describes briefly two experiments that have shown that a degree of Innateness and Introspection can be achieved, functionally, in robotic agents.**

Keywords - Autonomy; Innateness; Introspection; Machine Intelligence; Cognitive Architecture.

I. INTRODUCTION

Autonomous agents require cognition if they are to understand and adapt flexibly to complex worlds, and so underpin their ability to manage themselves. This is particularly important when the worlds are dynamic, uncertain and impossible to anticipate fully *ab initio.* There is a well-known definition of cognition as:

"…all processes by which the sensory input is transformed, reduced, elaborated, stored, recovered, and used..", [1].

To get the full benefits of cognition in agents, it is desirable to identify what the potential benefits are and to study cognitive processes that are manifested in humans and other higher mammals. These can be incorporated in a cognitive architecture, and they offer fundamental capabilities that are needed for practical application in AC (Autonomous Computing) systems, such as those for surveillance systems, animal behaviour modelling systems, robots, software agent systems, and other infrastructural systems for computing.

In an AC system, these capabilities should be amenable to distribution across the whole system, but, at the other extreme, there are situations where a centralised agent covering a cluster of relatively fixed low-level system elements, rather than one monolithic 'self', is appropriate. The structures and applicability of the alternative dispersion patterns is an area for further investigation. Our ideas cover the distributed scenario, but we focus on a relatively centralised situation as a starting point, and leave the distribution aspects for future elaboration.

Many functional elements are needed for autonomy in the general case. We argue that two specific capabilities, Innateness and Introspection, are needed for full cognition, and are often overlooked completely or under-cooked by researchers in AI (Artificial Intelligence) who offer cognitive architectures that could be considered for adoption in AC systems. A list of 'normative' capabilities that would be considered as 'cognitive' has been fairly well established in the literature. These are usually included in published cognitive reference models or cognitive architectures, and we believe that they should be considered by anyone seeking to produce AC systems that can deal flexibly and effectively with streams of signals and other inputs from their worlds. They include memory handling, various (somewhat subjectively selected) functions required for 'intelligent behaviour', and means of interacting with the environment. We claim that they must be supplemented.

For illustration of our approach in this introduction we use the relatively comprehensive LRMB (Layered Reference Model of the Brain) [2] for cognitive systems which, has been put forward specifically for use in AC systems. This model can be used either to help in explaining fundamental 'natural' cognitive mechanisms and processes [3 – 4], or to simply gather together specifications of capabilities that could be useful when engineering artefacts for various activities. Our focus here is on the latter use.

By common consent there are a lot of cognitive processes in 'natural intelligence'. The LRMB designers list 39 of these at six layers known as the sensation, memory, perception, action, metacognitive, and higher cognitive layers.

The designers of LRMB and other researchers taken collectively have produced a close-to-exhaustive list of features that should be possessed by any cognitive agent. There are various cognitive architectures [5] that have been mooted to capture the basis of cognition. They are intended to specify domain independent infrastructures for intelligent systems.

Starting from such suggestions, we desire to identify and understand some particular mechanisms and interactions for use in complex processes such as those requiring AC. However we find that they do not adequately cover the features on which we focus, i.e., on the Innateness relationships and interactions between what are called in LRMB the 'inherited and the acquired' cognitive functions, as well as Introspective processes that support deep reasoning. These features tend to be understudied in other reference models. For example according to Wang, in LRMB, these capabilities are relatively simple: the analogue of the operating system and applications in a computing system, particularly a real-time system and there is little mention of isolating a way of 'keeping an eye on' what is transpiring during cognitive activity, as a significant subsystem.

In this paper, our hypothesis is that consideration of cognitive architectures can greatly enhance AC systems, and we outline two novel aspects of a cognitive architecture that builds on previous work in this area, and make it particularly well suited to AC. This paper is therefore laid out as follows: we discuss the useful reference point or baseline of minimal cognition briefly, identifying the most basic mechanisms needed if a system is to be called cognitive. After this we define a general cognitive structure where we look at representative architectures in the literature and discuss the Innateness and Introspection functionality. We argue that these features are essential for true cognition. Basic learning methods to be applied in this research are considered and briefly demonstrated in an example of an exercise involving instincts. Finally, the paper discusses implications for measuring autonomy.

II. Minimal Cognition

Simple devices, which do not have functions that are commonly associated with cognition, such as reasoning and learning, can produce seemingly complex behaviour – using a stimulus response mechanism. Such mechanisms have been investigated by Konrad Lorenz [6], considered to be the father of Ethology. They have been termed 'fixed action patterns' or alternatively 'innate release mechanisms' where specific stimuli will trigger a fixed response. These events may appear intelligent but in fact are simple pre-programmed codes that do not deviate from pattern. An engaging example is given by Sharkey [7] where apparently conscious and smart behaviour can be observed with simple reaction mechanisms. The observed behaviour can be interpreted in many anthropomorphic ways, but the operation of the device is very simply explained by these limited, somewhat 'brainless' responses to sense data.

There are many important questions arising from thought experiments like this and related studies. We focus on one: When does a system become intelligent? However the *gedanken* above suggests a more fundamental preliminary question – one that we are actively studying in our labs. What is the minimal structure that is needed if we are to have a cognitive system? An alternative to the definition mooted earlier is to say that agents 'that reason act, perceive, and learn in changing, incompletely known, and unpredictable environments' are cognitive. The device in Sharkey's thought experiment clearly does not qualify.

Now, a two-component signal transduction (TCST) system [8] a molecular sensorimotor system in bacteria, has been discovered relatively recently, and this is seen by some to mark a boundary for cognition. The TCST system is important because it elucidates a molecular mechanism for adaptation and memory. Its sensorimotor organization is still dependent on metabolic activity, but it is 'organizationally autonomous', and functionality similar to that of the nervous system is claimed for it.

We claim that a minimally cognitive system must have two particular features before it can be called cognitive – Innateness and a degree of self-awareness, or Introspection. These features are discussed later.

III. Generalised Cognitive Architecture

As stated above, in research in this area at least two 'flavours' are identifiable…. 'Modelling invariant aspects of human cognition' – explaining/matching psychological phenomena; and 'an effective path toward building intelligent agents' – generating intelligent behaviour

Baars with the GWT (Global Workplace Theory) [9] and Moreno et al with CERA (Consciousness and Emotional Reasoning Architecture) [10] give examples of the types of architectural frameworks that are needed for a computational model of consciousness.

The first of these mainly deals with what Moreno calls A-Consciousness – accessibility of contents of memory for reasoning volition and speech. It seems to aim primarily at understanding consciousness in organisms. For CERA, Moreno includes inner perception or introspection (M-consciousness) and self-recognition and reasoning about the self (S-Consciousness) in his Reasoning consciousness. The functioning of artefacts based on CERA is important; however a goal of that model that we do not aim for is to get close to computational correlates of biological neural structures. Another architecture that is very well developed is ACT-R [11], and it presents a comprehensive list of functionality that would largely be agreed by any researcher working in this domain: Sense functions for visual and other sense processing; Motor functions for action; Memory functions for, e.g., short-term buffers and a long-term memory.

In such architectures, 'soft computing' functions are also needed, as are intentional functions for goals etc., along with a coordinator. The main components of 'Soft Computing' [12] (in Zadeh's SC Institute UC Berkeley) are:

- Fuzzy logic (FL)
- Neural network theory (NN)
- Probabilistic reasoning (PR)

Our emphasis is on exploration. We seek integrated systems for intelligent 'agental' behaviour, rather than piece-wise improvement of individual functions/modules. We want to accommodate the following generalised functionality: Perception and Action (motor) - outer stratum of Fig 1; Reasoning/Predicting/Deciding/Learning – second stratum, Soft (Computing) Functions', of Fig 1; Remembering/Learning - short or long memories STM (Short Term Memory) and LTM (Long Term Memory), in Fig 1. The general architectural framework for cognition presented in outline in Fig 1 gives the 'shape' of a sort of consensus of many contributors to this topic. In addition to showing the 'common consent' framework, we use it to distinguish our own architectural focus.

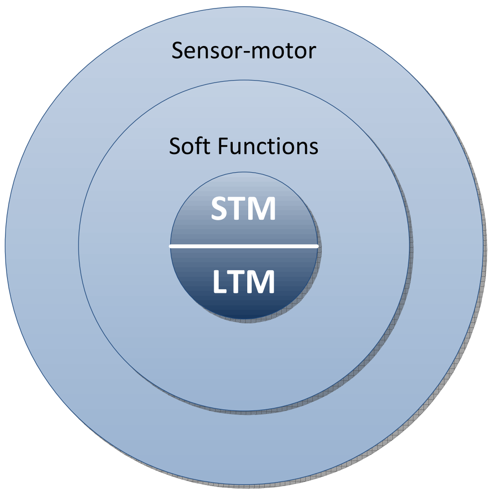


Figure 1. Standard Schema for a generalised Cognitive Architecture.

IV. Learning, Reasoning, and Uncertainty and Inconsistency Handling Mechanisms

Our research is approached from the direction of data and knowledge engineering. Our particular interest is in the areas of inconsistency handling and to an extent, machine learning. Both of these areas should arguably be key focuses for AC. A variety of projects have led to development of and usefulness of artefacts with this sort of functionality e.g., Medical, Transportation, Education, General Engineering, Telecoms, Music, Manufacturing, and Biology. Important topics include transitional (esp. time series) mining, causality, categorisation, reinforcement learning, and harmonisation issues, especially those connected with the obtaining of new knowledge.

The latter topic is of particular interest here – it is the primary catalyst which triggered the present proposals. As indicated above, it is all treated very much from an engineering viewpoint. It has led quite naturally to the contemplation of the possibility and practicability of adding a self-conscious aspect of agents to support ‘deep’ reasoning and thereby facilitate autonomous behaviour and AC.

To manage the changes of agent’s beliefs, we need to consider ways of revising or updating an agent’s current beliefs when new knowledge/evidence is obtained. To achieve this, a success principle must be maintained which states that new knowledge should be retained, and the minimal change principle is crucial. It argues that the agent’s prior knowledge should also be retained as much as possible while maintaining consistency. When the new knowledge is not guaranteed to be kept, merging has to take place to determine a new belief set based on the strengths of the prior beliefs and new evidence. There is much research [13 – 20] on various aspects of ‘Soft Computing’ relevant to this work on revising and measuring the amount of conflict and agreement between prioritized knowledge bases, and resolving the conflicts in such knowledge bases e.g., by lexicographic aggregation, combining them by negotiation, or merging them under various constraints.

Revision in numerical related theories, such as probability theory is handled differently. In Probability theory, revision is done by Bayes’ updating rule or Jeffery’s rule. In other theories, such as the Dempster-Shafer theory and possibility theory, the counterparts of Jeffery’s rule or Bayesian updating rule have been developed. The key idea of belief revision and merging in an agent environment is to accommodate new knowledge and to reach a consistent set of ne beliefs for the agent. In machine learning, methods are available for ‘Rough’ computations, discovering causal patterns in data through mining, reinforcement learning and feature subset selection based on relevance.

V. Specific Cognitive Architecture

The standard model of Fig 1 is supplemented by two important ingredients in our scheme [21] – functions and memory modules for each of two capabilities – viz: Innateness and Introspection (“Observer”).

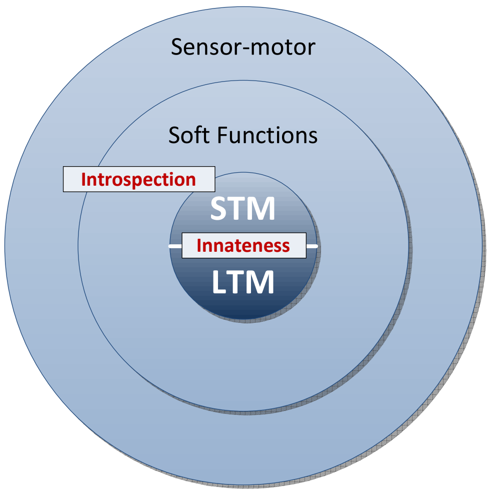


Figure 2. Cognitive Architecture Schema.

A. *Innateness*

A key factor in many programmes where flexibility is required of agents is what is built-in *ab initio*. Instinct is innate ability of agent to detect/react to/associate stimuli from the environment or from internal urges. By it an agent can begin, for example, to form de facto categories [22] (*“We are sensorimotor systems who learn to sort and manipulate the world according to the kinds of things in it, and based on what sensorimotor features our brains can detect and use to do so.”*), and thus learning. It can then use the categories and other learning outcomes to plan and predict, and to some extent modify the built-in innate reactions. Some behaviour can still be explained as innate, but the agent can learn and solve problems related to ‘goals’ – maybe, in some organisms, even ‘let its hypotheses die in its place’.

In an attack-flee scenario in one of our projects which we conduct using Khepera platform [23], two innate instincts are: Investigate (E) / Beware (B). Both of these trigger the collection of sense data (S) from which the robot learns. When a new object (e.g., a light) appears – both E and B lead to S and dominate when enough data is available, learning is possible, and this takes place systematically.

The decision resulting from a particular episode of exploration indicates which of E or B results – essentially whether the robot investigates further (E) or flees (B). This result depends on the evidence acquired according to 2 rules (illustrations are given in Rules 1 and 2 below) – from sense data.

Rule 1 - If light then B and S (strength .90) (an 'innate' rule) ….this says that if a light is detected, the robot has an instinct to be careful, but to passively collect data on the light's behaviour.

Rule 2 - If test is positive then E and S (a learned/acquired rule) ….this says that if the robot tries some test, such as: approach the light and look to see if object reacts aggressively, in which case the robot is put off (score -0.5 if yes); alternatively if the light does not react, score 1, and the robot is positively prompted to investigate further. Nothing is added otherwise.

There are two phases in episodes captured as behavioural traces. The first is illustrated in the table below – the result depends simply on the rules (instincts) and the sense data. At the end of this phase, when some termination criterion is reached, the balance of evidence lies in some particular direction e.g., Beware (as here) with certain strength (such as 0.90), probably indicating that the object is dangerous. The termination criterion for the tabled data is when the accumulated 'score' exceeds 5.

TABLE I. TRACE OF ACTION: STEPS IN A BEHAVIOURAL EPISODE

Agent Actions	**Object Actions**		
Approach	***Attack***	***React***	***Score Added***
1	0	0	1
1	0	0	1
1	0	1	-0.5
1	0	0	1
1	0	0	1
1	0	1	-0.5
1	0	0	1
1	0	1	-0.5
1	0	0	1

Just one piece of evidence is considered here – there could be others. Evidential reasoning can then be used to come to a decision. It is important to distinguish situations where this does not necessitate a particular decision - e.g., strength is only 0.90 - especially one which is against the purpose of the robot. This is in contrast to a more general purpose – such as to 'scientifically' reflect 'the real world'. Such a decision should be taken as a suggestion, but not a necessitation [24]. In the second phase, not illustrated here, further episodes could be conducted to get more persistent knowledge.

At the end of the episode in Table 1, we are at the point in the trace where the accumulated score exceeds 5 (so B dominates here). This pattern of robot plasticity or flexibility of behaviour could be applied in many other AC application areas such as surveillance, software agent systems and other complex computational systems. There is no requirement for self-awareness or Introspection in this example. We add the observer functionality/module as below.

B. Introspection

In previous work [25 – 26], the main theme of the simulation was to show grounding of real world experiences in an agent's sense data and hence some aspects of the common intuition of "understanding". The mapping of this sense data to internal mental constructs, categorising experience and patterns, and most importantly to justify actions, is considered to be a basis for understanding.

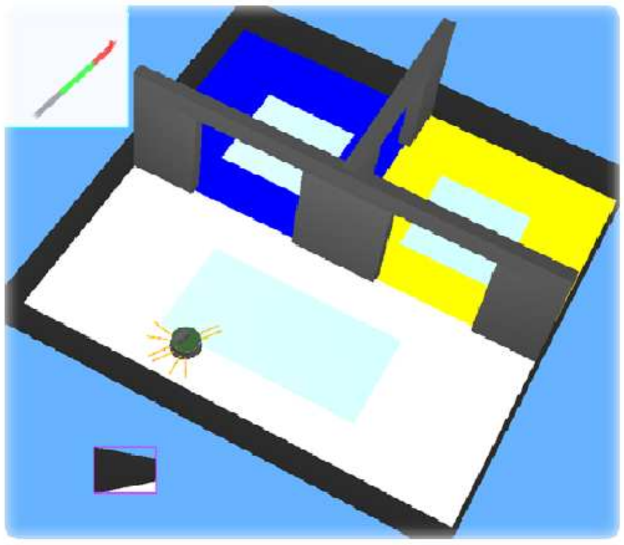


Figure 3. A simulated experiment in Webots, 3 rooms representing different activities an agent can engage and a colour trace top left that can be linked by an 'Observer' to a mood.

It is important to consider what understanding is. For present purposes it is the ability to rationalise ideas through abstractions in order to form a concept, and later to justify any resulting actions, in a given environment. A key consideration here is to ground concepts in sense data. In 'playback' systems the agent carries out actions which are pre-programmed, with no initial understanding of what it is doing. In our scenarios this is their situation initially and actions can be considered here as primitive "instincts". Concurrently there is a "mind" (Observer) linking these actions to an internal construct by abstraction. By doing this it can be said that body and mind have been separated function-wise to achieve this understanding.

Here we suggest that a level of understanding can be achieved by means of symbolic grounding. Two cognitive sub-agents called actor and observer are implemented to allow deep reasoning/query response in simple environments. The actor subagent pursues a programme involving engaging in activities and changing mood in a manner depending on the activities and their ordering. The observer sub-agent records the trace of activities and moods and links the activities and moods to its own sense data using a simple symbolic grounding function. For example, the actor from time to time enters rooms while leaving a colour trail. This colour trail changes depending on the room. The actor can be interrupted from time to time and this is recorded to see what the relationship it has with mood and activity pattern. Colours are linked to moods and rooms to activities. As the actor 'experiences' a particular path, the observer is able to classify, reason and explain what the actor is doing and experiencing in terms of sense data and summaries thereof. See Fig 3.

The environment is a working space which can be varied – e.g., from very simple rooms up to rather complex labyrinths, with various complexities of event mixes. The agent can pick up attributes of the environment form sensors and with the help of a built-in observer, answer factual (related to direct sense data values) and explanatory (related to summaries, etc. of sense data) questions about what it experiences. An example of patterns /characteristics of an agent are : while the activity is reading the mood is initially good, musing turns it to fair and discussion has a bad effect on mood.

This grounding can again extend the flexibility of artefacts such as the common AC applications, this time by relating 'understanding' to the basic inputs from the agent's environment.

VI. TOWARDS THE MEASUREMENT OF AUTONOMY

From an engineering point of view, it is important to be able to compare AC systems with respect to their degrees of autonomy. Like the closely related topic of intelligence tests, this is as yet not well understood and part of our work is to look at this issue. We do so from various angles here.

Keedwell [27] presents an initial proposal of staged testing for Intelligence, known as the Staged Developmental Machine, based on staged testing methods for developing children, per Piaget's theory. An analogous staged test device can be envisaged for AC systems. This proposed test offers a scalar value measure rather than the Yes/No of the Turing test, and there is no requirement for Natural Language Processing. The idea is that 'machines could be judged by their effective stage'. For use in testing for autonomy levels here, we highlight the following stages/part stages, per Keedwell, that we expect our artefacts to attain...

Stage 1 Sensory Perception
- Reacts to Basic Stimuli
- Understands Cause and Effect – predicts next step…
- Understands concept of objects (those controllable and those not) and what to expect from them.
- Uses trial and error to learn about the World (experimentation).

Stage 2 Pre-Operational
- Responding to (NOT Language understanding) relating to self (e.g., answering questions on state).
- Relating objects (though NOT via language) though not in current perceptual field (memory).

Stage 3 Concrete Operation
- Conservation of volume etc. of objects(e.g., estimating quantity of objects in visual field)
- Classification of objects logical rather than on attribute basis (animals/shapes…)
- Sorting objects (e.g., size or colour)
- Effect of Reverse of action (undoing) predictable

Stage 4 Formal Operations
- Ability to create hypotheses/experiments
- Abstract thought – prediction of interactions of objects in novel ways

We also propose [21] a staged approach to the measurement of degree of intelligence or autonomy via a similar scale based on complexity of the environment similar to the Sphex test [28], as well as a degree of Innateness and a degree of Introspection.

VII. SUMMARY AND CONCLUSION

This research is part of a project which focuses on developing a robot explorer. A more generally applicable, unique cognitive architecture has been proposed that can underpin wider AC functionality. It is expected that most autonomous agents at some point will encounter uncertain 'territory'.

Two key and somewhat distinctive features – Innateness and Introspection - are included in the architecture. These are considered to be indispensible if the flexibility and adaptability required for applications requiring autonomy is to be supported. We will use a probabilistic/possibilistic approach combined with classification methods to establish (inexact) rules and tools for AC. We will develop novel methods of evaluation for the added functionality. In particular, as an exemplar of dynamic application systems where the additional capabilities described here are targeted, a robotic agent will be able to explore and describe environments effectively.

REFERENCES

[1] Neisser, U. *Cognitive Psychology*. New York: Appleton-Century-Crofts, 1967. Print.

[2] Wang, Y. 2007, "Toward Theoretical Foundations of Autonomic Computing", *International Journal of Cognitive Informatics and Natural Intelligence*, vol. 1, no. 3, pp. 1-16.

[3] Langley, P., Laird, J.E., and Rogers, S. 2009, "Cognitive architectures: Research issues and challenges", *Cognitive Systems Research*, vol. 10, no. 2, pp. 141-160.

[4] Langley, P. and Choi, D. 2006, "A unified cognitive architecture for physical agents", *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI Press, pp. 1469-1474.

[5] Newell, A. *Unified Theories of Cognition*. Cambridge, MA: Harvard UP, 1990. Print.

[6] R.W. Burkhardt Jr., Konrad Lorenz, In: Michael D. Breed and Janice Moore, Editor(s)-in-Chief, Encyclopaedia of Animal Behavior, Academic Press, Oxford, 2010, pp. 298-303.

[7] Sharkey, N.E. and Heemskerk, J.N.H. 1996, "The Neural Mind and the Robot", *Neural Network Perspectives on Cognition and Adaptive Robotics* IOP Press, pp. 169-194.

[8] Van Duijn, M., Keijzer, F., and Franken, D. 2006, "Principles of Minimal Cognition: Casting Cognition as Sensorimotor Coordination", *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, vol. 14, no. 2, pp. 157-170.

[9] Baars, B.J. In the Theatre of Consciousness; Global Workspace Theory, A Rigorous Scientific Theory of Consciousness, *Journal of Consciousness Studies*, **4** (4), 1997, pp. 292-309.

[10] Arrabales, M. R. and Sanchis de Miguel, A. 2008, "Applying machine consciousness models in autonomous situated agents", *Pattern Recognition Letters*, vol. 29, no. 8, pp. 1033-1038.

[11] "About ACT-R". Act-R: Theory and Architecture of Cognition. 2011. March 24, 2011, http://act-r.psy.cmu.edu/about/

[12] "BISC Program; Soft Computing". The Berkeley Initiative in Soft Computing. 2009. March 24, 2011, http://www.eecs.berkeley.edu/~zadeh/

[13] Qi, G., Liu, W., Glass, D.H., and Bell, D.A. 2006, "A split-combination approach to merging knowledge bases in possibilistic logic", *Annals of Mathematics and Artificial Intelligence*, vol. 48, no. 1-2, pp. 45-84.

[14] Qi, G., Liu, W., and Bell, D.A. 2006, Merging stratified knowledge bases under constraints, *Proceedings of the 21st American National Conference on Artificial Intelligence*, (AAAI06): pp. 281-286.

[15] Qi, G., Liu, W. and Bell, D. 2010, "Measuring conflict and agreement between two prioritized knowledge bases in possibilistic logic", *Fuzzy Sets and Systems*, vol. 161, no. 14, pp. 1906-1925.

[16] Bell, D.A., Guan, J.W., and Lee, S.K. 1996, "Generalized union and project operations for pooling uncertain and imprecise information", *Data & Knowledge Engineering*, vol. 18, no. 2, pp. 89-117.

[17] Guan, J.W., Guan, Z., and Bell, D.A. 1997, "Bayesian probability on boolean algebras and applications to decision theory", *Information Sciences*, vol. 97, no. 3-4, pp. 267-292.

[18] Bell, D.A. and Guan, J.W. 1998, "Computational methods for rough classification and discovery", *Journal of the American Society for Information Science*, vol. 49, no. 5, pp. 403-414.

[19] Wang, H., Bell, D., and Murtagh, F. 1999, "Axiomatic Approach to Feature Subset Selection Based on Relevance", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 271-277.

[20] Guan, J.W. and Bell, D.A. 1993, "A generalization of the dempster-shafer theory", *Proceedings of the 13th international joint conference on Artificial intelligence - Volume 1* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 592-597.

[21] Bell, D. 2011, "The CogArch Framework", Unpublished paper.

[22] Harnad, S. "To Cognize is to Categorize: Cognition is Categorization". Categorization in Psychology. 2003. June 22, 2010, Proc UQAM Summer Institute in Cognitive Categorisation, http://www.ecs.soton.ac.uk/~harnad/Temp/catconf.html

[23] "Khepera Platform". K-Team Corporation | Mobile Robotics. 2011. March 24, 2011, http://www.k-team.com/

[24] An, Z., McLeish, M., Bell, D.A., and Hughes J, (1993) "How Did the Tiger Rumble the Donkey?" Proc 6th FLAIRS Symposium, Florida, pp136-141.

[25] Jin, Z. and Bell, D.A. 2003, "An experiment for showing some kind of artificial understanding", *Expert Systems*, vol. 20, no. 2, pp. 100-107.

[26] White, C. and Bell, D. 2010, "Actions and observations: Demonstrating aspects of understanding in a simple world", *Towards a Comprehensive Intelligence Test (TCIT): Reconsidering the Turing Test for the 21st Century Symposium*, ed. Ayesh, Bishop, Floridi, Warwick, April 2010, pp. 24-27.

[27] Keedwell, E. 2010, "Towards a Staged Developmental Intelligence Test for Machines", *Towards a Comprehensive Intelligence Test (TCIT): Reconsidering the Turing Test for the 21st Century Symposium*, ed. Ayesh, Bishop, Floridi, Warwick, April 2010, pp. 28-32.

[28] Esperjo-Serna J.C. 2010, "Connecting the Dots my own Way: Sphex-test and Flexibility in Artificial Cognitive Agents", *Towards a Comprehensive Intelligence Test (TCIT): Reconsidering the Turing Test for the 21st Century Symposium*, ed. Ayesh, Bishop, Floridi, Warwick, April 2010, pp. 1-6.

A Self-diagnosis Algorithm Based on Causal Graphs

Jingxian Lu
Orange Labs
2, Avenue Pierre Marzin
22300 Lannion Cedex, France
jingxian.lu@orange-ftgroup.com

Christophe Dousson
Orange Labs
2, Avenue Pierre Marzin
22300 Lannion Cedex, France
christophe.dousson@orange-ftgroup.com

Francine Krief
LaBRI - University Bordeaux 1
351, Cours de La Liberation
33405 Talence Cedex, France
francine.krief@labri.fr

***Abstract*—The concept of autonomic networking has been introduced to facilitate network management that is increasingly complex. It uses a closed control loop proposed by the autonomic computing. Obviously, this approach relies on a distributed architecture. This implies that the diagnosis algorithms have to be distributed in order to satisfy autonomic architecture requirements. In this paper we describe novel algorithms regarding the use of causal graph model to perform diagnosis distribution in telecommunication networks.**

***Keywords*—diagnosis algorithm; causal graph; distribution; self-diagnosis.**

I. Introduction

With the growth of network complexity and heterogeneity, the need for managing and controlling the network effectively by reducing human intervention seems essential. An autonomic system can monitor itself and adjust to the changing environment. It manages itself and becomes autonomic without human interventions [1]. The four distinct purposes of an autonomic system are: Self-configuring, Self-healing, Self-optimizing and Self-protecting. To achieve those purposes, autonomic systems have a detailed knowledge of their internal state as well as their environment through a continuous monitoring of eventual changes that could affect their components. Detecting changes helps the autonomic system adjust its resources and by monitoring, it continues to determine if the new measures satisfy the desired performance. This is an overview of the closed control loop (Figure 1) of self-management systems.

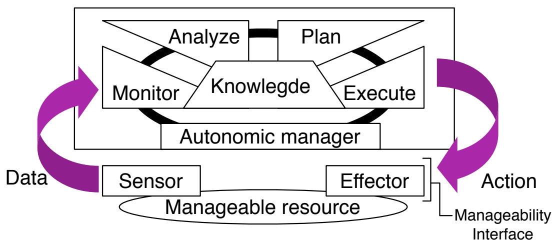


Fig. 1. *The closed control loop for Autonomic computing*

ANA [2] has defined the organization of the autonomic network with the possibility for an autonomic element to exchange information with different neighbor elements. The project FOCALE [3] has defined the ability of an operator to manage an autonomic network with business objective. A new control loop is used to manage the closed control loop of each network element. Because operators don't accept the black box (i.e., closed control loop) solution, an autonomic control loop must be validated by Service/Network Provider before integration in the network.

Self-healing means the system ability to examine, find, diagnose and repair system malfunctions [4]. With the development of autonomic network, the concept of self-healing is becoming more and more important for operators in order to satisfy customers needs. The need for dependable autonomic network has motivated researchers over the recent decades to investigate self-diagnosis. Self-diagnosis is an important part of self-healing. It is required that network could detect faults, handle alarms, execute related test by itself. The purpose of our studies is to introduce self-diagnosis in Self-healing function by autonomic networking.

This paper describes a novel algorithm regarding the use of causal graph model for performing self-diagnosis in telecommunication networks. We have presented the causal graph model in the paper [5]. The causal graph based diagnosis is a kind of model-based approach relied on a graph which represents causal propagation between causes and effects [6]. It could represent a process at a high level of abstraction and could be refined in a very detailed way if necessary. It allows to split the model into small parts and so to have an incremental way of modeling. By the way, it allows a very flexible way to model the knowledge. By definition, the causal graph is an acyclic graph. And it is composed of two parts: nodes representing partial states of the system and arcs representing causal relations. The causal graph structure is based on five types of nodes and one type of arcs (shown in Figure 2):

Primary cause is also called default or failure, and has no predecessor in the graph.

Intermediate cause represents partial states of the system. Intermediate and initial causes can't be directly observed.

Observation is the observable effect of failures and corresponds to alarms. In the graph, these nodes are leaves

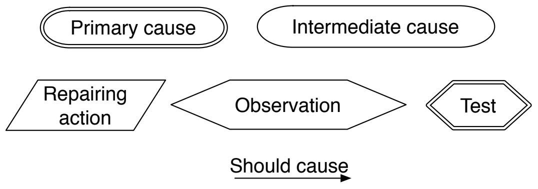


Fig. 2. *The components of causal graph*

and have no successors.

Test is used to verify a state or a hypothesis. From the view of causality, they are exactly as observations, the only difference is that they are not automatically notified, but need to be explicitly requested. (This node type could be also used to request a database.)

Repairing action is a special node, because it is not a partial state, and it doesnot take part in the causal relation between causes and effects. We can consider it as a label, it only serves when the diagnosis has been completed. These actions are considered as an additional treatment to repair.

Should cause means that the cause systematically leads to the effect (or the effect is a systematical effect of the cause). For example, "disconnected Livebox" *should cause* "disconnected IP".

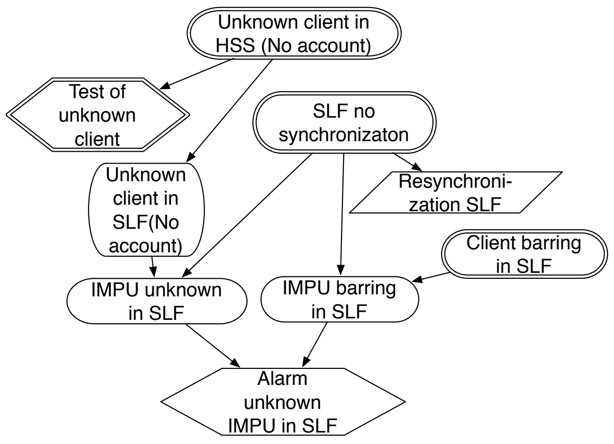


Fig. 3. *An example of a causal graph*

Figure 3 gives an example of causal graph in IMS environment. In this example, the alarm indicates that user's IP Multimedia Public Identity (IMPU) cannot be found in Subscriber Locator Function (SLF). And "Barring" means that a client account exists but has expired. The provisioning of the different databases is not simple because SLF and HSS are duplicated and they need to be updated at the same time with no failure. A non-synchronization failure or an update failure could happen. We refer the reader to [5] for more details. In fact, compared with [5], we have updated, here, some definitions and rules in order to improve and establish the algorithms.

The diagnosis process consists in collecting all the current observations, and then, determining all the primary causes which could explain these observations. Some tests could be launched during diagnosis in order to discriminate remaining ambiguities. Finally, the *repairing action* nodes connect to identify causes which determine the action to execute for repairing. To achieve the process, each node has some associated states:

Guilty: for a symptom node (test or observation), it means that the symptom was observed and should be explained by, at least, one of its predecessors. For an intermediate node, it means that the corresponding pathological state is asserted and, for a primary cause, it means that this cause is responsible for the current problem.

Innocent: for a symptom node, it means that the symptom is not observed (i.e., an alarm is not present or a test returns a negative result), for a cause primary or not, it means that it is not involved in the causal path and the cause should be searched elsewhere.

The diagnosis algorithms relies on a hypothesis set and are step-by-step procedures. The purpose is to build all the diagnosis by exploring hypothesis.The hypothesis is based on summation of state of causal graph nodes. Meanwhile, the hypothesis must strictly respect the following rules in order to be consistent.

R1: if a node is guilty, then all its successors are guilty.

R1': if a node is guilty, at least one of its predecessors should be guilty (except if the node is a primary cause).

R2: if a node is innocent, then all its predecessors should be innocent.

R2': if all the predecessors of a node are innocent, this node should be innocent.

R3: for a *test* node, its state will be *Guilty* or *Innocent* according to its result of execution (in other words, the execution of a test enables to add the observations in process).

Repairing actions connected to a guilty node of consistent hypothesis are candidates to repair diagnosed problem.

In next section, we will present the definitions needed in order to establish the base of algorithm theories and the formalized diagnosis algorithm.

II. Definitions and Notations

This section will present required definitions. The model is formally defined as follows.

Definition 1 (Causal graph): A causal graph G is an acyclic graph and consists in couple $(\mathbb{N}, S)$, where:

- $\mathbb{N}$ is a set of nodes from the causal graph;
- S is the causal relationship between these nodes.

Definition 2: For a causal graph $G = (\mathbb{N}, S)$, and for each node $n \in \mathbb{N}$, we denote:

- $S(n)$: the subset of successors of node n;
- $\widehat{S}(n)$: the subset of $\mathbb{N}$ containing all the descendants of n (transitive closure of S)
- $S^{-1}(n)$: the subset of nodes whose successor is n;
- $\widehat{S}^{-1}(n)$: the subset of all the ascendants of n.

As an extension, if $N \subset \mathbb{N}$, we note $\widehat{S}(N)$ the set of all the descendants of nodes N:

$$\widehat{S}(N) = \bigcup_{n \in N} \widehat{S}(p) \quad \text{and} \quad \widehat{S}^{-1}(N) = \bigcup_{n \in N} \widehat{S}^{-1}(p)$$

The hypothesis of acyclic graph is then translated by the following property:

Proposition 1 (Acyclic Graph): The causal graph $G = (\mathbb{N}, S)$ is an acyclic graph, if and only if

$$\forall n \in \mathbb{N}, n \notin \widehat{S}(n)$$

Definition 3 (Primary causes and Observations): For a given causal graph $G = (\mathbb{N}, S)$, we define the set of failures $\mathbb{P}$ and the set of observations $\mathbb{O}$ as follows:

- $\mathbb{P} = \{n \in \mathbb{N} | S^{-1}(n) = \emptyset\}$
- $\mathbb{O} = \{n \in \mathbb{N} | S(n) = \emptyset\}$

Note that $\mathbb{O}$ contains the nodes corresponding to alarms as well as tests.

Definition 4 (Symptom of a failure): An observation a ($a \in \mathbb{O}$) is an symptom (e.g., an alarm) of the failure $p \in \mathbb{P}$, iff there is a causal path from p to a (i.e., $a \in \widehat{S}(p)$). The symptoms set of a failure p is thus defined by $\widehat{S}(p) \cap \mathbb{O}$.

Definition 5 (Independance of failures): If two failures p_1 and p_2 are independent, then all the effects (observable symptoms and intermediate causes) of co-occurrence of these two failures is the union of effects of each failure. In other words:

$$\widehat{S}(\{p_1, p_2\}) = \widehat{S}(p_1) \cup \widehat{S}(p_2)$$

Definition 6 (Diagnosis): Let a causal graph $G = (\mathbb{N}, S)$ and a set $\mathcal{A} \subset \mathbb{O}$ of alarms (observations), a diagnosis D is a subset of $\mathbb{P}$ which can explain the set of observations. In other words:

$$D \text{ is a diagnosis for } \mathcal{A} \text{ iff : } \quad \mathcal{A} \subseteq \widehat{S}(D) = \bigcup_{p \in D} \widehat{S}(p)$$

Note that if D is a diagnosis for $\mathcal{A}$, then $\forall p \in \mathbb{P}$, if there is no interaction between the failures (hypothesis of the independent failures), $D \cup \{p\}$ is a diagnosis for $\mathcal{A}$.

Proof: If D is a diagnosis for $\mathcal{A}$, then we have $\mathcal{A} \subseteq \widehat{S}(D)$. Under the hypotheses of independent failures, we have $\widehat{S}(D \cup \{p\}) = \widehat{S}(D) \cup \widehat{S}(p)$. If $\mathcal{A} \subseteq \widehat{S}(D \cup \{p\})$, then $D \cup \{p\}$ is a diagnosis for $\mathcal{A}$.

For the same set of symptoms $\mathcal{A}$, you can order the corresponding diagnosis using the relation $\subseteq$. We will rather find the minimal diagnosis than the direct diagnosis.

Definition 7 (Minimal Diagnosis): Given a causal graph $G = (\mathbb{N}, S))$ and a set $\mathcal{A} \subset \mathbb{O}$ of the alarms (observations), a diagnosis D is minimal diagnosis for $\mathcal{A}$, if and only if, there is no diagnosis D' for $\mathcal{A}$ when $D' \subsetneq D$. In other words, if D and D' are two minimal diagnosis for $\mathcal{A}$ and $D' \subset D$, then $D = D'$.

The objective of diagnosis will be to find all the minimal diagnosis.

Definition 8 (Hypothesis of diagnosis): A hypothesis H of diagnosis is defined by:

1) a set $\mathcal{G}(H) \subseteq \mathbb{N}$ of "guilty" nodes;
2) a set $\mathcal{I}(H) \subseteq \mathbb{N}$ of "innocent" nodes;
3) a set $\mathcal{X}(H) \subseteq \mathcal{G}(H)$ containing all unexplained nodes (i.e., without guilty predecessor).

A hypothesis is closed when it explains all its symptoms and remains consistent, that is to say when:

$$\mathcal{X}(H) = \emptyset \quad \text{and} \quad \mathcal{I}(H) \cap \mathcal{G}(H) = \emptyset$$

Definition 9 (Fusion of hypotheses): Consider two diagnosis hypotheses H and H', let us define the fusion of these two hypotheses $H \otimes H'$ as follows:
$\mathcal{G}(H \otimes H') = \mathcal{G}(H) \cup \mathcal{G}(H')$
$\mathcal{I}(H \otimes H') = \mathcal{I}(H) \cup \mathcal{I}(H')$
$\mathcal{X}(H \otimes H') = (\mathcal{X}(H) \setminus \mathcal{G}(H')) \cup (\mathcal{X}(H') \setminus \mathcal{G}(H)) \cup (\mathcal{X}(H) \cap \mathcal{X}(H'))$

III. Diagnosis Algorithms

In this section we propose the base of the implemented diagnosis algorithm and the various proposed extensions to take into account the tests during the current diagnosis. We first introduce a centralized algorithm in this section, and then develop the distributed algorithm in the next section.

A. Diagnosis Search

The algorithm of calculation $\mathcal{D}$ relies on a set of hypothesis $\mathcal{H}$ and ends when there is no node to explain in the hypotheses (i.e., all hypotheses are closed). During initialization of the algorithm, we start from the initial hypothesis H_0 which will be based on $\mathcal{A}$ as all the set of alarms to explain.

Algorithm 1 Calculation of the set $\mathcal{D}$ of diagnosis

```
Require: $\mathcal{H} = \{H_0\}$ with $H_0$ defined by $\mathcal{G}(H_0) = \mathcal{X}(H_0) = \mathcal{A}$ and $\mathcal{I}(H_0) = \mathbb{O} \setminus \mathcal{A}$
1: while $\mathcal{H} \neq \emptyset$ do
2:   Choose $H$ in $\mathcal{H}$ (and delete it from $\mathcal{H}$)
3:   if $\mathcal{X}(H) = \emptyset$ (the hypothesis $H$ is closed) then
4:     The diagnosis $D$ is defined by $D = \mathcal{G}(H) \cap \mathbb{P}$
5:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{D\}$
6:   else
7:     Algorithm 2: Develop $H$
8:   end if
9: end while
```

We can modify the policy of diagnosis progress by changing the procedure of choice H in $\mathcal{H}$ of the diagnosis hypothesis to develop (line 3 of the algorithm 1).

Algorithm 3 Propagate m in H'

```
1: if $\widehat{S}(m) \cap \mathcal{I}(H') \neq \emptyset$ then
2:   return $H'$ is inconsistent
3: else
4:   $\mathcal{G}(H') \leftarrow \mathcal{G}(H') \cup \widehat{S}(m)$
5:   return $H'$ is consistent
6: end if
```

Algorithm 2 Develop H of minimal diagnosis

```
1: Choose $n \in \mathcal{X}(H)$ (and delete it from $\mathcal{X}(H)$)
2: if $S^{-1}(n) = \emptyset$ or $S^{-1}(n) \cap \mathcal{G}(H) \neq \emptyset$ then
3:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{H\}$ ($n$ is already explained, $H$ remains valid)
4: else
5:     for all $m \in S^{-1}(n)$ do
6:         Create $H'$ (a duplicate of diagnosis hypothesis $H$)
7:         $\mathcal{G}(H') \leftarrow \mathcal{G}(H') \cup \{m\}$
8:         $\mathcal{X}(H') \leftarrow \mathcal{X}(H') \cup \{m\}$
9:         execute Algorithm 3: Propagate $m$ in $H'$
10:        if H' is consistent then
11:            $\mathcal{H} \leftarrow \mathcal{H} \cup \{H'\}$
12:        end if
13:        $\mathcal{I}(H) \leftarrow \mathcal{I}(H) \cup \{m\}$
14:    end for
15: end if
```

Algorithm 2 avoids exploring redundant number of hypothesis and can lead to minimal diagnosis.

B. Extension of absent alarms or lost alarms

An absent or a lost alarm corresponds to *an absence of information*, we don't know if the symptom is present or not.

By default, algorithm 1 supposes that if an alarm is unobserved, it is not active (i.e., $\mathcal{I}(H_0) = \mathbb{O} \setminus \mathcal{A}$). If a subset $\mathcal{A}' \subset (\mathbb{O} \setminus \mathcal{A})$ of alarms are absent, then the previous algorithm is initialized with $\mathcal{I}(H_0) = \mathbb{O} \setminus (\mathcal{A} \cup \mathcal{A}')$ and the algorithm remains the same.

The obtained diagnosis assumes that some of these alarms could be active but lost. For a given diagnosis D, the set of active alarms is: $\bigcup_{p\in D} \widehat{S}(p) \cap \mathbb{O}$. And so, the alarms assumed as lost by this same diagnosis are: $\left(\bigcup_{p\in D} \widehat{S}(p) \cap \mathbb{O}\right) \setminus \mathcal{A}$.

C. Extension of tests

We want to handle the case where certain observations don't show spontaneously but correspond to tests to be executed. In this case, one of certain observations $\mathbb{O}$ are labeled as tests (belongs to $\mathbb{T}$).

The algorithm 4 is the main body of the algorithm, and it will rely on the following procedures: Algorithm 5 and Algorithm 6. Note that, for a test node, the algorithm 6 will replace the algorithm 3 to execute.

In the algorithms, $\mathcal{T}^*$ means the test before the execution or the test waiting for the execution. $\mathcal{T}^+$ and $\mathcal{T}^-$ are the results after test execution, here $\mathcal{T}^-$ stands for a positive result (Innocent) and $\mathcal{T}^+$ for a negative one (Guilty).

The test choice policy (line 1 of the algorithm 5) could be based on the cost induced by the test (in terms of delay, of computer resource, etc.).

The development of an algorithm can go through the selection of executed tests.

Algorithm 4 Calculation of the set $\mathcal{D}$ of diagnosis

```
Require: $\mathcal{A} \cup \mathcal{A}' = \emptyset$
1: $\mathcal{T}^* = \mathcal{T}^+ = \mathcal{T}^- = \emptyset$
2: $\mathcal{H} = \{H_0\}$ with $H_0$ defined by $\mathcal{G}(H_0) = \mathcal{X}(H_0) = \mathcal{A}$ and $\mathcal{I}(H_0) = \mathcal{A}'$
3: while $\mathcal{H} \neq \emptyset$ do
4:     Choose $H$ in $\mathcal{H}$ (and delete it from $\mathcal{H}$)
5:     if $\mathcal{G}(H) \cap \mathcal{T}^- \neq \emptyset$ ou $\mathcal{I}(H) \cap \mathcal{T}^+ \neq \emptyset$ then
6:         $H$ is consistent (and abandoned as in contradiction with tests executed in other hypothesis)
7:     else
8:         if $\mathcal{X}(H) = \emptyset$ then
9:             if $\mathcal{T}^* \cap \mathcal{G}(H) = \emptyset$ then
10:                (the hypothesis $H$ is terminated and without waiting)
11:                The diagnosis $D$ is defined by $D = \mathcal{G}(H) \cap \mathbb{P}$
12:                $\mathcal{D} \leftarrow \mathcal{D} \cup \{D\}$
13:            else
14:                execute Algorithm 5: Launch a test for $H$
15:            end if
16:        end if
17:    else
18:        if $\mathcal{T}^* \cap \mathcal{G}(H) = \emptyset$ (No tests for $H$) then
19:            execute Algorithm 2: Develop $H$
20:        else if We choose to launch a test for $H$ then
21:            execute Algorithm 5: Launch a test for $H$
22:        else
23:            execute Algorithm 2: Develop $H$
24:        end if
25:    end if
26: end while
```

Algorithm 5 Launch a test for H

```
1: Choose and execute a test $t \in \mathcal{T}^* \cap \mathcal{G}(H)$
2: if The result of $t$ is a "detected problem" then
3:     $\mathcal{T}^+ \leftarrow \mathcal{T}^+ \cup \{t\}$
4:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{H\}$ ($H$ is consistent)
5: else
6:     $\mathcal{T}^- \leftarrow \mathcal{T}^- \cup \{t\}$
7:     (we abandon $H$ which is in contradiction with the result of test)
8: end if
```

IV. Distribution of the algorithm

A distributed system is a set of components that interact by exchanging messages through a communication network. When detecting an anomaly, components emit alarms for monitoring the system.

For a distributed architecture: each local diagnosis system first calculates a diagnosis using only local information, then a global diagnosis is determined by communication between each local diagnoser. This approach has the advantage of a complete distribution of the diagnosis task. It seems also interesting to make the diagnosis as close as

Algorithm 6 Propagate m in H'

```
1: if $\widehat{S}(m) \cap (\mathcal{I}(H') \cup \mathcal{T}^-) \neq \emptyset$ then
2:     return $H'$ is inconsistent
3: else
4:     $\mathcal{G}(H') \leftarrow \mathcal{G}(H') \cup \widehat{S}(m)$
5:     $\mathcal{T}^* \leftarrow \mathcal{T}^* \cup \left( \left( \widehat{S}(m) \cap \mathbb{T} \right) \setminus \mathcal{T}^+ \right)$
6:     return $H'$ is consistent
7: end if
```

possible to the source of alarms, because it causes fewer problems with delays and lost alarms. Nevertheless there is a risk of generating a large number of communications. Some approaches focus on issues related to distributed algorithms, and propose distributed monitoring methods based on several communicating supervisors [7][8][9]. Other approaches combine distributed algorithms with the characterization of system behaviors [10][11][12][13][14]. The present algorithms belong to this second category.

A. The principle

The distributed algorithm for the autonomic networks is based on network architecture: the different nodes of causal graph correspond to various components states of autonomic systems (network, IMS component, etc.).

The principle is to cut the causal graph according to components of system architecture. Each autonomic component (or group of components) has its own diagnoser that will establish its diagnosis hypotheses according to its own causal graph (subgraph of the global causal graph) and will compare to the other neighbor diagnosers in order to get a compatible and consistent diagnosis.

The availability of distributed algorithm could be assured by the two following points:

- the local diagnosis is compatible with the local observations and the local causal subgraph.
- all the local diagnosis should be compatible with at least one local diagnosis of all its neighbours.

B. Algorithm

The local diagnosis is based on a subgraph $G' = (\mathbb{N}', S)$ of $G = (\mathbb{N}, S)$, where the set $\mathbb{N}' \subset \mathbb{N}$ and the relation S applied on $\mathbb{N}'$. The symptoms of the local diagnoser are: $\mathcal{A} \cap \mathbb{N}'$.

Some nodes of $\mathbb{N}'$ are connected with the other nodes which aren't in $\mathbb{N}'$ (not in this local graph).These nodes are called "connection nodes" between the local diagnosers. The change of diagnosis hypotheses of these "connection nodes" will trigger message exchange between diagnosers.

Definition 10 (Boundary of effects and of causes): Given a causal graph $G = (\mathbb{N}, S)$ and a sub-graph $G' = (\mathbb{N}', S)$ de G, where $\mathbb{N}' \subset \mathbb{N}$.
The boundary of effects in G' is defined by:

$$\mathbb{F}_{G'} = \left\{ n \in \mathbb{N} \middle| n \notin \mathbb{N}' \text{ et } \left(S^{-1}(n) \cup \mathbb{N}' \right) \neq \emptyset \right\}$$

The boundary of causes in G' is defined by:

$$\mathbb{F}^{-1}_{G'} = \left\{ n \in \mathbb{N}' \middle| S^{-1}(n) \not\subset \mathbb{N}' \right\}$$

We assume the network architecture is hierarchical, which enable be share the task of diagnosis between different levels of abstraction of a given system.

Distributed algorithm

Assume a partition of $\mathbb{N} = \mathbb{N}_1 \cup \cdots \cup \mathbb{N}_N$, and so a set of local subgraph $G_i = (\mathbb{N}_i, S)$, $\forall i, j,\ G_i \cap G_j = \emptyset,\ 1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N$. The distributed algorithm is based on messages exchanged between neighbor diagnosers: each subgraph sends and updates a message $\mathcal{M}$ ($\mathcal{M} = \mathcal{H}$) for each neighbor. The idea is that: when G_i receives $\mathcal{M}$ from its neighbor G_j, then G_i fuses $\mathcal{H}_i$ with $\mathcal{H}_j$ (if two hypotheses are coherent), and then G_i will recalculate $\mathcal{D}$ to update it; if $\mathcal{D}$ is complete, G_i sends result of $\mathcal{D}$ to operator, creates at the same time an unique ID and sends a message $askE_{ID}$ to ask if its neighbors have finished their diagnosis. Related algorithms (Algorithms 7, 8 and 9) are the following:

Algorithm 7 Fusion of $\mathcal{H}_i$ and $\mathcal{H}_j$

```
1: for all $(H_i, H_j) \in \mathcal{H}_i \times \mathcal{H}_j$ do
2:     if $H_i \otimes H_j$ is coherent then
3:         $\mathcal{H}'_i \leftarrow \mathcal{H}'_i \cup \{H_i \otimes H_j\}$
4:     end if
5: end for
6: $\mathcal{H}_i \leftarrow \mathcal{H}'_i$
7: execute Algorithm 8: Diagnosis $\mathcal{D}$ in subgraph
```

Algorithm 8 Diagnosis $\mathcal{D}$ in subgraph

```
Require: $\mathcal{M} \leftarrow \emptyset$
1: while $\mathcal{H} \neq \emptyset$ do
2:     Choose $H$ in $\mathcal{H}$ (and delete it from $\mathcal{H}$)
3:     if $\mathcal{X}(H) = \emptyset$ then
4:         sends $H$ to operator
5:     else if $\mathcal{X}(H) \cap \mathbb{F}^{-1}_{G'} \neq \emptyset$ then
6:         $\mathcal{M} \leftarrow \mathcal{M} \cup \{H\}$
7:     else
8:         execute Algorithm 2: Develop $H$
9:     end if
10: end while
```

Algorithm 9 Termination of $\mathcal{D}$

```
1: if $\mathcal{M} = \emptyset$ then
2:     Creat an unique $ID$
3:     sends $askE_{ID}$ to its neighbor diagnoser
4: else
5:     sends $\mathcal{M}$ to its neighbor diagnoser
6: end if
```

In these 3 algorithms, we have not yet add the *test* and *repairing actions* in order to simplify the algorithms.

Termination of algorithm

For a distribution of algorithm, the termination of diagnosis should be considered. When $\mathcal{H} = \emptyset$, hypothesis of diagnosis is closed (i.e., diagnosis is complete). Then, all the results of diagnosis will be sent to operator to finally validate and execute the repairing actions.

Let us define a special message okE_{ID} as the response of $askE_{ID}$ to notice the agreement of "complete diagnosis" between the neighbor diagnosers and a message $cancelE_{ID}$ as a notice of deleting the finished diagnosis.

Algorithm 10 Reception $askE_{ID}$ from G_j

```
 1: if $H \neq \emptyset$ then
 2:     sends $H$ to neighbor diagnosers
 3:     delete $H$ from $G_i$
 4: else if $E_{ID} \in \mathcal{E}$ then
 5:     sends $okE_{ID}$ to $G_j$
 6: else
 7:     $\mathcal{E} \leftarrow \mathcal{E} \cup \{E_{ID}\}$
 8:     $\mathcal{W}(E_{ID}) \leftarrow$ {all the $ID$s of neighbor diagnosers
        except $G_j$}
 9:     $\mathcal{S}(E_{ID}) \leftarrow G_j$
10:     sends $okE_{ID}$ to all $\mathcal{W}(E_{ID})$
11: end if
```

Algorithm 11 Reception okE_{ID} from G_j

```
1: $\mathcal{W}(E_{ID}) \leftarrow \mathcal{W}(E_{ID}) \setminus \{G_j\}$
2: if $\mathcal{W}(E_{ID}) = \emptyset$ then
3:     sends $okE_{ID}$ to $\mathcal{S}(E_{ID})$
4: end if
```

Algorithm 12 Reception $cancelE_{ID}$ from G_j

```
1: sends $cancelE_{ID}$ to all $\mathcal{W}(E_{ID})$
2: $\mathcal{W}(E_{ID}) \leftarrow \emptyset$
```

In the algorithm 10, $\mathcal{E}$ is the set of ID of all the neighbors which send $askE_{ID}$. $\mathcal{W}(E_{ID})$ is the set of all the neighbors which have sent $askE_{ID}$ and wait for the response okE_{ID}. And $\mathcal{S}(E_{ID})$ is the source of diagnosis.

V. Conclusion and Future Work

We have introduced a novel algorithm regarding the use of causal graph model for performing self-diagnosis. To do so, we first introduced the notion of causal graph modeling. Mathematically, it's possible to derive diagnosis algorithm based on causal graph: local diagnosis are executed by local diagnosers and take the form of message passing algorithms. The keystone of the algorithms is to achieve a self-diagnosis in a global environment.

Future work will focus on the implementation of the self-diagnosis algorithm in an IMS platform to verify diagnosis performance, and then, according to the results of diagnosis, we will try to refine the policies of the algorithms and to improve it.

References

[1] IBM, *Autonomic Computing: IBM's perspective on the state of information technology*, Technical report, 2001.

[2] ANA: Autonomic Network Architecture, http://www.ana-project.org

[3] J. C. Strassner, N. Agoulmine, and E. Lehtihet, *FOCALE - A novel autonomic networking architecture.* pp.64-79, 2007.

[4] D. Tosi, *Research Perspectives in Self-healing Systems*, Technical Report of the University of Milano-Bicocca. July, 2004.

[5] J. Lu, C. Dousson, B. Radier, and F. Krief, *Towards an autonomic network architecture for self-healing in telecommunications networks*, in Proceedings of 4th International Conference on AIMS. Zurich, Switzerland, 2010.

[6] L. Console, D.T. Dupre, and P. Torasso, *A theory of diagnosis for incomplete causal models*, in Proceedings of IJCAI. USA, 1989.

[7] R. K. Boel and J.H. van Schuppen, *Decentralized Failure Diagnosis for Discrete Event Systems with Costly Communication between Diagnosers*, in proc. 6th Int. Workshop on Discrete Event Systems. WODES' 02, pp.175-181, 2002.

[8] R. Debouk, S. Lafortune, and D. Teneketzis, *Coordinated decentralized protocols for failure diagnosis of discrete event systems*, Discrete Event Dynamic Systems : theory and applications, vol. 10(1/2), pp. 33-86, 2000.

[9] T. Yoo and S. Lafortune, *A General Architecture for Decentralized Supervisory Control of Discrete-Event Systems*, Discrete Event Dynamic Systems: Theory and Applications. vol. 12(3), pp. 335-377, July, 2002.

[10] R. Su, W.M. Wonham, J. Kurien, and X. Koutsoukos, *Distributed Diagnosis for Qualitative Systems*, in proc. 6th Int. Workshop on Discrete Event Systems, WODES'02, pp. 169-174, 2002.

[11] Y. Pencole, M. Cordier, and L. Roze, *A decentralized model-based diagnostic tool for complex systems*, Int. J. on Artif. Intel. Tools. World Scientific Publishing Comp., vol. 11(3), pp. 327-346, 2002.

[12] S. Genc and S. Lafortune, *Distributed Diagnosis Of Discrete-Event Systems Using Petri Nets*, in proc. 24th Int. Conf. on Applications and Theory of Petri Nets. LNCS 2679, pp. 316-336, June, 2003.

[13] R. K. Boel and G. Jiroveanu, *Distributed Contextual Diagnosis for very Large Systems*, in Proceedings of WODES'04, pp. 343-348, 2004.

[14] E. Fabre, A. Benveniste, S. Haar, and C. Jard, *Distributed Monitoring of Concurrent and Asynchronous Systems*, Journal of Discrete Event Systems, 2005.

Convergence singularities: As man becomes machine and machine becomes man

Carl Adams
School of Computing, University of Portsmouth, UK
Portsmouth, UK
e-mail: carl.adams@port.ac.uk

Abstract—. **Within the AI domain there is discussion on the singularity event, the situation where machines achieve a level of consciousness and intelligence equal to or exceeding that of humans. This paper explores different perspectives on such a singularity. The main contentions are that a) it is a moving target in that using technology man is also expanding human capabilities – as such it is a convergence than a one sided movement; b) it is not one singularity but an array of singularities representing multiple attributes of humans and; c) singularities should be considered in a wider system perspective that takes account of organizational structures, individual motivations and practicalities. This paper provides an Information Systems and autonomic systems perspective on a machine-man/man-machine singularity. The paper hopes to make contribution by extending the discourse on such singularities and raise some practical and philosophical questions for society and policy decision makers.**

Keywords-man-machine singularity;convergence singularity

I. INTRODUCTION*HEADING 1)*

The 'singularity' is the expected situation where artificial intelligence (AI) machines achieve a level of intelligence and consciousness equal to or exceeding that of humans. This concept has engaged many imaginations over many decades in both academic research and in popular science fiction literature and film (e.g. 2001 A Space Odyssey, I Robot, Terminator, Matrix). Indeed some of the main themes, concepts and implications of machine intelligence and consciousness outstripping humans are often explored in fiction and film better than in academic research: Academic research usually focuses on specifics of the 'how to do' elements of machine intelligence (such as new algorithms and technologies) whereas fiction explores wider sets of issues and takes forward particular 'what if' scenarios.

However, the research endeavours into AI have been phenomenal [18], possibly tying into some underlying human desires for technology and advancement [15]. In the early computing years the possibility of machines achieving human level intelligence started to become a possibility as technological advances took place, though considerable challenges we identified:

"In the future, if and when this degree of complexity is attained by a single computer, we shall perhaps be able to find out if it can be made to match the power of thought of the human brain.

The problem of programming such a super-computer will still remain, however. The limitations of man as a programmer will always, in the end, set a limit to the intelligence that can be simulated by a machine. A computer's 'artificial intelligence' is prescribed by man, and a higher intelligence is demanded for the prescription than for the execution. Man, as the originator, will always be on top." [12]

Over the more recent decades considerable effort within the computer science domain has been focused on developing AI techniques and technologies many of which explore ways of surmounting the 'originator barrier', including Logic systems, Bayesian networks, Neural networks, Simulated annealing, Swarm Intelligence algorithms and many more [18] . At the same time as developments in AI the processing and storage power of computer systems have expanded at phenomenal rates over the decades providing a raw processing intelligence base for running such AI systems. Concurrently with these advancements, research into consciousness has also been fairly active drawing upon several fields including computer science, neuroscience, cognitive science, psychology, philosophy and others [5],[3]. Seemingly the foundations for developing machines that 'think' are well in place [14],[15] and that we are well on the way towards a singularity event.

A wakeup call to the imminent singularity was the chess match in May 1997 between the world chess champion Garry Kasparov and the Deep Blue chess-playing super computer developed by IBM. Deep Blue won a six-game match (two wins to one with three draws – though Kasparov claimed IBM cheated and asked for a rematch, which was refused by IBM who relatively quickly dismantled Deep Blue). In a very limited sense this was an example of a singularity event: a machine beat the best of humans in a complex thinking task – that of playing chess.

Issues and concerns over a singularity event have been gaining ground in the wider public and academic communities over the previous few years with emergence of specialist conferences, forums and workshops focusing on the possible singularity event. Possibly the most prominent of these being the singularity summit in the US, which aims to: "explore the rising impact of science and technology on society. The summit has been organized to further the understanding of a controversial idea – the Singularity scenario" (from http://www.singularitysummit.com/). Tied in with the summit is the Singularity Institute (see http://singinst.org/) which aims to raise awareness, engage in research and initiate public debate on the singularity event.

The singularity event is one of the big issues in future of computer science as well as the wider communities within society: A singularity event is likely to affect many stakeholders who may interact with and use such systems. Probably one of the strongest contenders for a route to machine consciousness is the development of sophisticated autonomic computer systems [2]. Such autonomic systems have to operate in real time in dynamic environments without human intervention. This paper explores an Information Systems (IS) and autonomic perspective on such singularities. The paper tries to extend existing singularity discourse along three main themes:-

a) The singularity is a moving target in that the capabilities of man are also expanding, both in evolutionary sense and also in the adaptation and employment of drugs and technologies. Consequently the singularity it is more a convergence than a one sided movement. Man's capabilities are being extended moving towards capabilities found in a range of technologies and machines (this we refer to as man-machine) while machines are moving towards man's capabilities, such as empathy and consciousness (this we refer to as machine-man).

b) It is not one singularity but an array of singularities representing multiple attributes of humans and of autonomic system machines. For instance, human attributes cover socializing, empathy, creativity (and other) capabilities as well as intelligence and consciousness capabilities. Different as well as overlapping sets of attributes are likely to emerge from the development of sophisticated autonomic systems.

c) Singularities should be considered in a wider organizational and system perspective that takes account of forces of change, organizational structures, individual motivations, economics, information flows and needs, practicalities and many other attributes affecting the success of a technology. Any likely future sophisticated autonomic systems that evolve into some form of consciousness will not operate in a vacuum: such autonomic systems are likely to interact with other entities (human and/or machine) and there is likely to be some economic, business, social or other imperative driving such interaction.

As such, this paper extends the discourse on man-machine singularity than found within the existing literature by combining an IS and autonomic systems perspective. The paper argues that such perspective is needed to make sense of the issues and implications of heading towards man-machine/machine-man convergence singularities.

II. A MOVING TARGET: MAN TO MACHINE AND MACHINE TO MAN

This section examines the convergence of extending capabilities – those of the expanding capabilities of man (man-machine) and those of intelligent machines (machine-man). The concept of singularity becomes non static; indeed it may even be asymptotic if the capabilities of man subsume the capabilities of machine intelligence. First we will examine the man-machine activity then the machine-man activity.

A. Man to machine (Man-machine)

Man has also used tools and technology to improve and extend human capabilities, indeed tool making is one of the main attributes of man that distinguishes us from other animals and beings [4],[16],[9]. For instance, the use of throwing sticks and bow and arrows have been used to extend human capabilities in throwing weapons.

Much of the use of technology has been to address weaknesses or the loss of capabilities such as when people grow old. For instance glasses and hearing aids (hearing trumpets to digital hearing aids) have been used to improve sight and hearing capabilities - to bring them back to a 'normal' level for people. The support routes of technology development for people with medical conditions or disabilities often provide a base to improve human capabilities. Wheel chairs used to help people that can't walk enable wheel chair users to go faster than the normal walking pace or running pace of people that can walk (e.g. a 70 year old can trundle along at 8 mph in their electric wheel chair - far faster than a normal 70 year old can go without technology aid; a wheel chair marathon runner can often go faster than a normal marathon runner).

There is a wide range of areas where technology has been used to enhance human capabilities. Man has used telescopes or microscopes to increase sight capabilities well beyond normal levels. Hearing or sensing of sound has been extended from very low frequencies through to high frequencies heard by bat. One could even include sensing all the way up through the spectrums extending perception through higher and higher frequencies to light and X-rays. Cars, boats, submarines, trains, airplanes and spaceships have extended human capability to move around enabling humans to travel faster, carry bigger loads and travel through more environments than the human body can alone.

The man-machine mix can be classified into non-invasive and invasive. The non-invasive mix of technology with humans has been taking place for centuries, and would include all the traditional use of tools and technologies (e.g. hand tools, cars, production machinery). The non-invasive mix of technology would also include ICT, computing technologies and the Internet. These enable humans to communicate with each other almost instantly from around the globe as well as provide access to massive amounts of information. This also comes with sophisticated searching and processing capabilities which combined enables a potential step leap in cognitive capabilities providing understanding of and interaction with the environment from the micro to the macro level.

The non-invasive extension of human capabilities, the man-technology mix, provides the base for a closer interaction between man and technology - the invasive extensions (i.e. once a technology has been produced it can be applied closer to the body functioning). For instance, invasive heart pace-makers bring normal heart functioning to people with heart conditions. External hearing aids can be replaced with ear implants and so providing a more robust, finely tuned and less obvious support for people with hearing difficulties, as well as addressing hearing problems that cannot be addressed by an external hearing aid. Artificial limbs can be used to enable leg amputees to walk again. Much invasive use of technology has been to address deficiencies and weaknesses in people, but they can also be used to extend capabilities beyond the normal. It is only a small step to improve ear implants to be able to provide extended hearing range well beyond the normal levels. Paralympics runners can use carbon fiber limbs to run faster than normal runners.

This is the realm of the 'six million dollar man' and other science fiction. However, the reality has caught up with many of the themes in fiction. In David Rorvik’s [17] 1970’s book, ‘As man becomes machine’, the concepts of integrating machinery and computers with human biological functioning are well discussed, based on the current science and research of the time. Though Rorvik’s book seems a little dated now, the concepts of Electronic Stimulation of the Brain (ESB) and Bio-feedback Training (BFT) show the possibilities of closer interaction between people and technology.

More recently, the work by Professor Kevin Warwick, at Reading University’s Department of Cybernetics provides an up to date indication of where such technology is going. Professor Warwick – classed as the first cyborg – experiments in the late 1990’s involved implanting a microchip into the nervous system into his arm and used it for direct communication with a computer to control simple machinery, like switching on and off a light. These pioneering experiments involving the neuro-surgical implantation of a device that can directly interact with the median nerves of his left arm allowed a direct link between his nervous system and a computer. A follow on set of experiments involved having a similar device connected up to his wife then to communicate directly from one person to another directly via the nervous systems.

Interaction and control directly from human nervous system is now a reality. From past technological development it seems that the military and intelligence communities are usually at the forefront of using ‘useful’ technology, possibly using the technology 5 to 10 years before it gets to the general public. ‘Useful’ technology in a military context can include anything that could have a possible military purpose. For instance, it may be very desirable for generals to be able to locate and communicate with each troop on the ground, and to be able to receive high definition reconnaissance data from each of those troops. It would be advantageous for troops on the battlefield to be able to get better information on where their colleagues are, the terrain and obstacles around them and of course information about where the enemy is. Equally, it is easy to see where it would be desirable to have location, health and ‘condition’ information on troops particularly when it comes to casualties. Much of the future of man-machine enhancements may already be here.

B. Machine to man (Machine-man) Machine Consciousness and Autonomic Systems

Two aspects of machine-man will be explored here, one focusing on autonomic computer systems and the other on machine consciousness. One of the main sets of trends in computer systems has been towards more powerful and more sophisticated machines that undertake more functionality. The result of this has been computer systems that have to deal with more autonomy, increasing functionality and consequently increases in operating uncertainties and inconsistencies. This is the realm of autonomic computer systems. Ganek and Corbi [10], in their much quoted paper discussing the ‘dawning of the autonomic computing era’ describe the main attributes of autonomic computing systems as being self managing systems with self-configuring, self-healing, self-optimizing and self-protecting capabilities. Ganek and Corbi’s view of autonomic systems come from a business systems perspective, such as large network operating systems within complex and dynamic environments. However, autonomic computer systems derive their name from biology based on autonomic nervous systems (also known as involuntary nervous systems) within the human body, and are essentially the regulatory mechanisms of digestion, respiration, circulation of the blood etc. Although biology autonomic systems cover mostly involuntary functions, such as heart rate, these can be brought under some voluntary control (for instance, conditioning and meditation to control heart rate). There is much debate within medical and biology spheres on the relationship between the apparent voluntary and involuntary control mechanisms within the body: "The fact that much behaviour is involuntary and conscious raises such questions as: Why is some behaviour voluntary, and under conscious control? It seems that high rates of information-processing in unusual situations require consciousness, and are voluntary" [11, p66].

As a set of body operating functions becomes well defined with interrelated feedback systems, then this set of functions can operate within some autonomic control system [2]. Within that autonomic control system there are involuntary controls operating almost independently of the voluntary controls. However, as the amount of information and processing increases for functions, and variety in operating situations increase, then this requires more voluntary control mechanisms. Applying this biology analogy to computing autonomic systems it seems likely that some form of ‘voluntary’ control will be required as the amount of information and variety in operating situations increases. Machine-man capabilities can be extended by taking an autonomic approach, using a mix of voluntary and involuntary functionality, with the voluntary capabilities operating at a higher abstract level. This is getting closer to thinking [15] – a very human attribute.

Closely related to this concept of machines thinking is machine consciousness. Of course consciousness is a complex phenomenon that occupies much debate within the sciences and philosophy. Susan Blackmore's [5] work collates together conversations from some of the leading philosophers, computer scientists and neuroscientists on what constitutes consciousness, along with the possibility of generating consciousness in machines.

As Aleksander [3] "the strategy for designing conscious machines is tough but, in the end, doable. It is founded on unraveling the undoubtedly complex functioning of the brain and transferring this understanding into computational machinery." (p13) and defines his own personal five tests, or axioms, for being conscious (i.e. to distinguish between appearing to be conscious and being conscious):

"The five axioms, the five different kinds of thought which are important to me and I feel need distinguishing are the following:

1) I feel that I am part of, but separate from an 'out there' world.
2) I feel that my perception of the world mingles with feelings of past experience.
3) My experience of the world is selective and purposeful.
4) I am thinking ahead all the time in trying to decide what to do next.
5) I have feelings, emotions and moods that determine what I do." [3,p34]

Aleksander brings out a common theme in discourse on machine consciousness that of consciousness equates to human consciousness. However, as with the discussion on autonomic systems above, perceptions within an environment must be based on the range and level of sensory inputs about that environment and the characteristics of the environment itself. Different environments and different sensory inputs may well results in different types of consciousness. Consider different animals, sharks (and other fish), dolphins, birds, bats, dogs – these all have different environments and different senses and sensory inputs to what people have. Any consciousness of these beings (if indeed they can be classed as being conscious – a philosophical topic that engages researchers in this area, but for this paper assume that they are conscious) would be different to human consciousness. Similar reasoning would apply to a machine-man consciousness being the product of a different set of sensory inputs from a similar by likely different environment (say a more electronic environment). Humans from their blinkered human perspective have an understanding of what consciousness means for them; other intelligent and conscious being (if they exist) may well have their own (and possibly equally blinkered) perspective of what consciousness means for them.

III. MULTIPLE SINGULARITIES

One of the contentions in this paper is that there is not one singularity but an array of singularities representing multiple attributes of humans and human intelligence. For instance, human beings have evolved over many years (millions of years) to be able to explore, sense, perceive, learn about, move within, interact and shape their environment. They have evolved an imagination capable of creating many forms of art and complex imaginary worlds and contexts in stories and film. They have also evolved to be very sophisticated social entities developing many different social structures from small family groups to formal and informal structures to enable multiple millions of people to live together in the same city or country. They have evolved other social structures to enable production and distribution of food, products and services (such as markets, money and bartering systems). They have evolved and developed many forms of communication and languages. They have also evolved to be very adaptive and innovative able to populate most areas of the planet (and even off the plant), and develop many different types of tools and technologies. Perhaps one of the most significant elements of human attributes, or higher order of human intelligence and consciousness, is the ability to be humane, to have empathy to others (of the same and other species) and the environment. This is moving from self awareness to self responsible.

Humans are temporally rich in skills, they can interpret and make sense of past events and look forward and make fairly accurate best guess predictions on future events (for instance predicting what another person or animal might do in a situation, or predict the weather or the next eclipse). People are quite good at learning and passing this learning on to other people. Even organized groups of people, say at an organizational level are able to learn. Humans are able to learn about learning and deal with multiple constructs both abstract and 'real' [8].

Each of these rich dimensions of humanness and human intelligence provide a dimension with which potential singularities can emerge. The man-machine/machine-man singularities should really be considered as a convergence and synergy of capabilities mixing man and machine attributes that best fits a particular context.

IV. ORGANIZATIONAL AND SYSTEM PERSPECTIVE OF SINGULARITIES

A further contention of the paper is that the technological (and human) evolution towards singularity events should be considered in a wider organizational and system perspective. Technology impacts people, organizations and society. There are many forces and influences impacting the development and adoption of new technology and new systems. Man-machine/machine-man singularities represent new states of technology development and understanding of that development requires some understanding of the underlying forces of change, the resulting impact on organizational structures, individual motivations and general practicalities. This is the realm of Information Systems.

New technological change impacts a range of stakeholders, organizations, working practices and social interaction – which can stimulate further technological change. For instance, Damsgaard and Gao [7] examining the evolutionary innovation of the mobile telecommunications market, describe the process as an 'innovation circle' where the introduction of a new technology stimulates further innovations in the mobile market place. Similarly [1] takes a similar approach that includes the user in the innovation process and refers to this as the problem-solution space or innovation-space. In a dynamic environment, when technology changes so too does user practices and needs. As user practices and needs evolve further opportunity emerges for technological led innovations. This dynamic environment, it is argued, consists of a changing problem-solution space which provides the ideal circumstances for innovation to germinate. When considering man-machine/machine-man singularities to this dynamic evolutionary development process, particularly in identifying the forces of change, the corresponding changes in working and social practices and the likely influences on further technological change.

A man-machine or machine-man entity would be resource intensive compared to say just a normal human (or machine). The extra resources of course would provide enhanced capabilities - enabling the entity to do more things, quicker, better or longer – but the resources consumed could be very significant. For instance the Garry Kasparov and Deep Blue chess match in 1997 involved a super computer, a team of programmers and computer engineers focusing on just that limited (but challenging) task. This raises issues of the economic case for development of technological man-machine/machine-man singularities. In what circumstances would such an enhanced entity be economically viable or sustainable? Would some man-machine/machine-man singularities be more economically, or practically viable than others, and if so then which ones?

Further, man-machine/machine-man singularities, than have to be considered in the context of the already existing inequalities in the world [19],[21],[6]: "About one sixth of the world's population – mostly in the rich countries – are lucky enough to live in relative wealth, and the rest – mostly in developing countries – live in relative deprivation, if not desperate poverty" [22, pv]. As [20] identifies, on one level there is unprecedented opulence and on the other there is remarkable deprivation and destitution. The enhanced entity of man-machine/machine-man will likely raise specter of new type of underclass. The rationale and motivations towards developing man-machine/machine-man singularities have to be considered within the inequalities in the world.

Further questions and issues emerge around the development of such enhanced man-machine/machine-man entities and systems. For instance there is still the 'originator barrier' of AI systems, particularly in understanding the full workings and functionality of such an advance piece of technology. There will clearly be challenges in identifying the information and performance needs to develop a conscious intelligent system, but there will also be a further set of needs to consider of the actual conscious intelligent entity itself. With man-machine/machine-man singularities the number of stakeholders to be considered has been increased by (at least) one.

V. Discussion and conclusion

This paper has tried to provide a more IS perspective on man-machine singularity than found within the existing computing literature. We argue this is important since an IS perspective is needed to make sense of the issues and implications of heading towards man-machine/machine-man convergence singularities. The discourse on the man-machine singularity has to move away from just the technology and philosophical questions: Questions also need to be raised on the practicalities of how such systems can be developed and applied and the corresponding impact on society, organizations and people. To help identify some of these important questions this paper has extended the man-machine singularities discourse by identifying: the singularity is a moving target (in that using technology is helping man expand human capabilities) so it is more a convergence or synergistic even; that the singularity is actually an array of different singularities representing the multiple attributes of what it means to be intelligent humans beings and; singularities should be considered in a wider system perspective than just the technological and academic – it should take account of the practicalities of introducing new technologies including the economic, organizational, social and individual impacts.

Considering these perspectives some key questions emerge including: How will society be organized with such enhanced beings? Would machine-man entities have the same status, roles, responsibilities (social, tax, voting rights) in society as man-machine entities, or normal people? What will be the legal implications of the evolvement of machine-man entities and how will laws change to recognize this? Which man-machine/machine-man singularities are more likely than others? Which singularities have more economic or social benefit for society, or which will be most problematic. What will be the optimal or practical level of man-machine/machine-man entities within a population? Would it be one to one or one to ten? What criteria and metrics would make sense in identifying an optimal level? Which of the man-machine/machine-man singularities are likely to be problematic (for humans) and which are likely to offer benefits? These are likely to be key philosophical questions that will task computer scientists and many other people in the years to come.

Would raw intelligence measures, say processing or storage power, be more significant (on what ever scale) than say social intelligence, such as empathy or social capacity, or creative intelligence? If the human race in the future encounters other intelligent species, say in space travels, then would good or enhanced empathy, or innovative skills be more beneficial than raw intelligence?

Is there a natural synergy between man and machine (coming from either the man to machine or the machine to man route) that provides some form of optimal being or existence (however that can be measured or categorized)? Evolution theories suggests that successful entities will evolve to fulfill a niche within the wider environment – in which case what would a man-machine/machine-man niche be?

As many commentators on AI and machines consciousness identify, designing conscious machines is 'tough' calling for many breakthrough steps in computer science. Equally, as this paper has hoped to show, the route to practical and useful conscious machines has many 'tough' challenges outside of computer science and indeed many tough challenges across disciplines.

For many of the questions raised here we can draw upon key thinkers throughout history that have grappled with similar social and ethical dilemmas. Would the world become an autocracy directed by one single uncontrolled supreme man-machine/machine-man being? This is similar to Plato's ideas of a society ruled by an enlightened few – the enhanced man-machine/machine-man entities. Technology could play a role in enabling that to happen with the increased power of electronic monitoring systems and processing power to make massively informed decisions on how societies can be run. However, there is a strong argument that this would not happen. As machine-man systems became more autonomic and conscious then they are likely to adopt other human attributes, such as being individuals and valuing systems similar to those of humans. Also, democratic systems and market systems seem to have evolved to be an efficient mechanism for allocation of resources between many groups of people as well as providing the stimulus for technological advancement. At a basic level one could say they we are fundamentally human animals with human animal needs, such as food, shelter, reproduction, socialization etc. Are our fundamental human needs likely to be that different over time? Maslow developed the concept of a hierarchy of needs, sometimes represented as a triangle of needs with basic food and survival needs at the bottom, leading up to self-actualization at the pinnacle. Is life, the fundamentals that is, likely to be that much different today than several decades ago when Maslow first introduced it? As machine advancement continues to mimic human capabilities then they may have their own hierarchy of needs, some of which may be similar to humans. Indeed, the more closely the machine-man/man-machine entities become then the closer the hierarchies of needs will become. They will be motivated by similar things and value similar things. But it is up to society in how the direction of technology unfolds. However the man-machine/machine-man singularities emerge and evolve the arguments covered in this paper indicate that this will be a rich area for future IS research.

References

[1] Adams C. (2007a) Can Innovation Survive Virtual Teams and Outsourcing? Cutter IT journal, Executive Updates, 01 September 2007, pp4-8. Available from http://www.cutter.com/content/innovation/fulltext/updates/2007/ieau0701.html, accessed 12/9/07.

[2] Adams C (2007b) Autonomic systems, coping strategies and dream functions. The Third International Conference on Autonomic and Autonomous Systems, ICAS 2007, June 19-25.

[3] Aleksander I. (2005) The world in my mind, my mind in the world: Key mechanisms of consciousness in people, animals and machines. Imprint Academic, Exeter, UK.

[4] Ardrey R. (1977) The social contract. Collins Press, Bungay, Suffolk, UK.

[5] Blackmore S. (2005) Conversations on consciousness. Oxford University Press.

[6] Collier P. (2008) The Bottom Billion: Why the Poorest Countries are Failing and What Can Be Done About It. OUP, Oxford.

[7] Damsgaard J. and Gao P. (2004) Analysing mobile telecommunications market development. IFIP TC8 working conference on Mobile information Systems, 15-17 September, Oslo, Norway. In Lawrence E., Pernici B. and Krogstie J. (2004) Mobile Information Systems. Springer.

[8] Deacon T. (1997) The symbolic species: The co-evolution of language and the human brain. Penguin, London.

[9] Dunbar R. (2005) The human story: A new history of mankind's evolution. Faber and Faber, London.

[10] Ganek A.G. and Corbi T.A. (2003) The dawning of the autonomic computing era. IBM Systems Journal, March, 2003.

[11] Gregory R.L. (Ed) (1987) The Oxford companion to the mind. Oxford University press

[12] Holingdale S.H. and Tootil G.C. (1965) Electronic Computers. Pelican, Harmondsworth, Middlesex, UK.

[13] Leakey R. and Lewin R. (1992) Origins reconsidered: In search of what makes us human. Little, Brown & co., London.

[14] McCorduck P. (1979) Machines Who Think. San Francisco: W.H. Freeman and Company.

[15] McCorduck P. (2003) Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence. CRC Press.

[16] Morris D (1969) The naked ape. Corgi, London.

[17] Rorvik D. (1973) As man becomes machine. London.

[18] Russell S. (1998) Artificial Intelligence: A Modern Approach. Pearson Education.

[19] Sen A. (2001) Development as Freedom. Oxford Paperbacks, Oxford.

[20] Sen A. (1999) Commodities and Capabilities. OUP, Oxford.

[21] Stiglitz J.E. (2003) Globalization and Its Discontents. OUP, Oxford.

[22] Stiglitz J.E. and Charlton A. (2007) Fair Trade For All: How Trade Can Promote Development. OUP, Oxford.

An Autonomic Security Mechanism Based on Novelty Detection and Concept Drift

Gesiel Rios Lopes, André L. V. Coelho, Raimir Holanda Filho
Master's Course in Applied Computer Sciences
University of Fortaleza - UNIFOR
Fortaleza - CE - Brazil
Email: gesielrios@edu.unifor.br, {acoelho,raimir}@unifor.br

***Abstract*—The growth in the number of elements and services in current computer networks and the integration of several technologies, in order to provide connectivity and services everytime and everywhere, have renderede the computing infrastructure complex and susceptible to malicious activities. In this sense, the use of autonomic computing principles arises as an alternative to face these challenges, in special to deal with the development of security mechanisms capable of infering malicious activities with the purpose of self-protecting the infrastructure against attacks.This paper presents and evaluates a novel model of an autonomic element which is based on the notions of novelty detection and concept drift. The element is able to infer malicious activities that may compromise the proper functioning of the network. Experimental results, extracted from real samples network traffic, indicate that the element autonomic element infact is very useful.**

***Keywords*—Autonomic network, autonomic element, novelty detection, concept drift.**

I. Introduction

In recent decades the industry has increasingly produced new technologies, products and services, thus satisfying the demands for computer systems with large storage capacity and fast communication. These advances, although they bring important benefits, they have produced highly complex and heterogeneous systems. As a result, ensure security, fault tolerance, design capabilities, integration and management are becomming critical factors [1]. Therefore, solving the problems and difficulties generated by this scenario is a big challenge, which will only be achieved when computer systems be designed and built to adjust to changing situations, treating and managing of their resources efficiently. These factors are the main motivations for developing a new management system, delegating such tasks to machines, thus enabling the removal of administrative staff from the cycle and putting it only in a supervisory position. This view has been referenced in the academic community as autonomic computing [2].

This paper addresses the challenge of implementing an autonomic element, with the following combination of properties: continuous unsupervised learning of new concepts, an approach based on statistical properties extracted from the initial packet payload of the TCP flows and the use of a sorting algorithm of low computational power, the clustering algorithm k-means [3]. Moreover, our work is not limited to submitting a proposal, because it evaluates an implementation of an autonomic element using real traces. It is expected that the autonomic element proposed in this work will enable self-protection, self-configuration and able to learn new concepts over time, based on flows generated by the traffic.

The paper is organised as follows: Section II presents the concepts of novelty detection, the continuous learning OLINDDA (OnLIne Novelty and Drift Detection Algorithm) algorithm used by the autonomic element proposed in this paper and the k-means clustering algorithm. Section III presents our proposed autonomic element. Section IV describes the procedure for collecting and labeling the data used in the validation of the autonomic element. Section V presents our results and analysis, and finally, on Section VI our paper is concluded with some remarks and future works.

II. Novelty Detection

The mining of data streams brings several challenges to machine learning techniques. Its importance increases with the need for real-time analysis of a large amount of data. In this scenario, the ability to identify emerging concepts is an important attribute, and the Novelty Detection technique can contribute toward that goal. In Machine Learning, novelty detection can be defined as the identification of patterns that differ somehow from those normally expected, keeping a certain similarity with the concept of outliers [4]. In the context of this work, we consider a novelty as a new concept, i.e., an abstraction of instances or examples that share a set of characteristics that differ from those previously identified.

If the characteristics of a new concept are similar to a concept learned initially we use the term concept drift. Within this paper, was used the OLINDDA, a cluster-based algorithm which considers the problem of detecting concepts from Internet Traffic and operates over a continuous flow of data [4].

A. The OLINDDA algorithm

In this subsection, we present the functions of the OLINDDA. In terms of the learning paradigm, it may be divided in two phases: a supervised learning phase, where a normal model is built from a set of examples that describe the data domain, and an unsupervised learning phase, where unlabeled examples arriving from a data stream are analized and novel concepts are continuously learned [4]. The implementation of OLINDDA used in the proposed autonomic element made use of the k-means clustering algorithm [5].

1) The k-means algorithm: Cluster analysis aims to divide the elements of one sample, or population, in clusters so that the elements belonging to the same group are similar to each other based on variables (features) used for measuring them, and the elements in different clusters are heterogeneous with respect to these same characteristics.

Let $x \in R^n$ be a vector of features with a known probability density p. We want to compute a partition of R^n, $\{R_1, R_2, \ldots, R_n\}$, representing each class by a vector $\hat{x}_i \in R_i$ that is called centroid. The choice of centroids and the partition is made to minimize the average distance of each vector to its centroid:

$$D = E\left\{d\left(x, r\left(x\right)\right)\right\} \tag{1}$$

The solution adopted in practice is to alternately estimate the centroid and the partition of space. The algorithm of Lloyd-Max is summarized in the following table [5]:

1) Initialization: Choose a static number of classes and initialize the centroids of each class according to some criterion.
2) Classification: Determine a partition of X, $\left\{X^1, X^2, \ldots, X^c\right\}$, associating each training sample to the class whose centroid is closest:
$$x \in X^k : k = \arg\min_i d(x, \hat{x}_i) \tag{2}$$
3) Update centroids: Calculate new centroids for each class by the equations
$$\sum_{x \in X_p} \frac{\partial}{\partial \hat{x}_p} d(x, \hat{x}_p) = 0, \qquad p = 1, \ldots, c, \tag{3}$$
where X_p means the data set assigned to the p-th class. If $d(x, y) = \|x - y\|^2$,
$$\hat{x}_p = \frac{1}{N_p} \sum_{x \in X^p} x, \qquad p = 1, \ldots, c \tag{4}$$
where $N_p = |X^p|$
4) Back to step 2 until there is a stop condition.

In the second step, each pattern is classified in the class of the closest centroid. During the third step, the centroids are recalculated, starting to occupy the midpoint of the standards in its class.

B. Distinction between normal and unknown

For construction of the normal concept, OLINDDA initially considers a set of examples that are used to represent the normal concept. The initial data are grouped into k clusters by the k-means clustering algorithm. For each cluster, we calculate the distance between the centroid and the farthest example. This allows us to establish a decision boundary for each cluster. The union of the boundaries of all clusters is the global decision boundary which defines the model. A new unseen example that falls inside this global boundary is consistent with the model and therefore considered normal; otherwise, it is labeled unknown. Otherwise, the example is marked as a member of an unknown profile and moved to a short-term memory for further analysis. That memory works like a FIFO queue to avoid its uncontrolled growth, eliminating old samples to permit the inserting of new samples [4].

C. Identification of novelty and concept drift

To monitor the formation of clusters in the short-term memory of unknown data, we use k-means to generate k candidate clusters. For a candidate cluster to be considered valid, which might indicate a concept in our approach, it must fulfill the following requirement. We use the in implementation of the proposed autonomic element, the sum of the squares of the distances between examples and the centroid divided by the number of examples as a validation criterion of the degree of cohesion, defined by:

$$d(x_j, \hat{x}_i) = \frac{\sum\limits_{x_j \in c_i} (x_j - \hat{x}_i)^2}{N_i} \tag{5}$$

where c_i represents the candidate cluster, $\hat{x}_i$ their respective centroids and N_i the number of examples of their respective centroids belonging in c_i.

Then, we compare this value to the mean distance between the centroid and examples of the candidate cluster. If the value for the candidate cluster is lesser than or equal to the one obtained for the model, the candidate cluster is considered valid. This restriction aims at selecting clusters whose density is not lower than that of the model. Once a candidate cluster has been validated: it may either represent a novel concept (novelty) or be an evidence that the normal concept is undergoing a change (concept drift).

To establish the limit between conceptual change and novelty, we calculate the global position of a centroid for the Normal template and then calculates the distance between this centroid and the overall centroid farther away, resulting in a decision boundary. If the centroid of the new cluster is beyond this boundary, the new cluster is considered a new and separate for further analysis. Otherwise, it is attributed to this new cluster a change of concept, and this information is used to update the standard model itself. An overview of the OLINDDA is shown in Figure 1.

III. OUR PROPOSED AUTONOMIC ELEMENT

In this section, we present our proposed autonomic element (Figure 2), which aims to identify, by monitoring the network traffic, malicious activity resulting from deviations and possible changes in the normal concept.

Our autonomic element uses the general format of autonomic element, proposed by [2] and based on traffic subflows [6], which can be defined as a stream of packet being transmitted between a pair of hosts. Only TCP traffic was analyzed during this research, generating a set of 52 statistical variables candidates for discriminant, calculated based only on information obtained from the packet headers (for example:

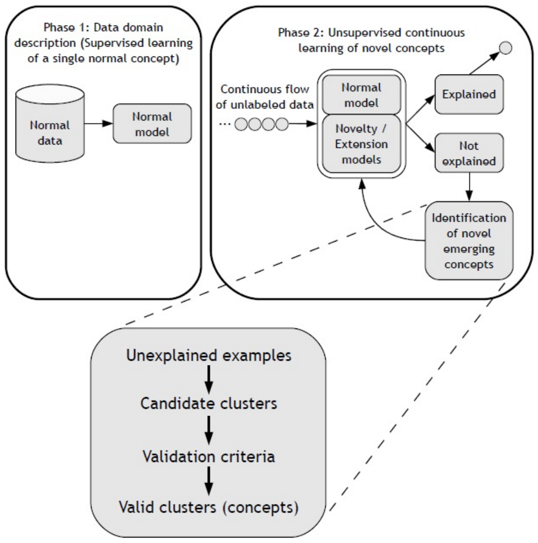

Phase 1: Data domain description (Supervised learning of a single normal concept)
Normal data
Normal model
Phase 2: Unsupervised continuous learning of novel concepts
Continuous flow of unlabeled data
Normal model
Novelty / Extension models
Explained
Not explained
Identification of novel emerging concepts
Unexplained examples
Candidate clusters
Validation criteria
Valid clusters (concepts)

Fig. 1. Overview of the OLINDDA [4].

TABLE I
EXAMPLES OF CANDIDATE VARIABLES TO DISCRIMINANT

Candidate Variable
Total of packet (Client to Server)
Minimum Windows Size (Server to Client)
Mean of Inter-Packet Length Variation(Client to Server)
Maximum of Bytes in Ethernet Packet (Client to Server)

packet size and TCP flags), without the use of inspection of payload and port numbers. The variables are calculated for each direction of a bidirectional flow (client to server and server to client). Table I shows some of these variables.

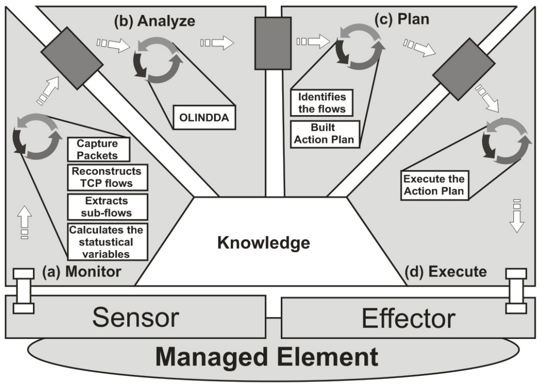


Fig. 2. Proposed Autonomic Element.

The implementation of the proposed autonomic element occurs in two phases: the first is called supervised phase and the second phase called unsupervised. In the first phase, the autonomic element determines the statistical variables that best discriminate the traffic considered normal(attack free) from a trace previously classified. At this stage the autonomic element use the mechanisms proposed to select the set of statistical candidates variables that best characterizes the normal traffic. Once you have determined the set of variables, the autonomic element builds the initial model, which will be used to identify potential malicious activities on the network and verifies the accuracy of the chosen variables. In the unsupervised phase, the proposed element performs the implementation of the autonomic cycle, as described below:

Monitor: The autonomic element (Figure 2(a)), works on the network traffic in promiscuous mode and reconstructs the TCP flows, extracts sub-flows, and calculates the statistical variables selected during the supervised phase, passing them to the analysis phase.

Analyze: Our autonomic element (Figure 2(b)), verifies if each sub-flow generated in the previous phase can be explained by the normal concept built during supervised phase. All sub-flows explained by the normal concept are labeled as normal to the next phase. The sub-flows not explained are temporarily stored in the short-term memory of unknown data. When the amount sub-flows stored in the short-term memory reaches a certain threshold, new clusters are generated, which will be labeled as a change of concept or novelty in accordance with the criteria used by OLINDDA and forwarded to the next stage.

Plan: Our autonomic element (Figure 2(c)), identifies the flows which were explained by the normal concept. These follow without any analysis in the next phase. The clusters labeled as concept drift or novelty are built into the normal concept. The information of the of remaining packets of sub-flows the clusters labeled as a novelty are stored, so they are discarded at the execute phase.

Execute: Our autonomic element (Figure 2(d)), in this phase execute the action plan established in the plan phase, i.e., all remaining packets of each sub-flows that were considered normal concept following without any changes and all the remaining sub-flows identified as novelty are discarded, because in our approach these packages represent some malicious activity.

Knowledge: stores the informations found by the autonomic element in all phases, such as the statistical variables, limits, the normal model, error rates, among other items.

Figure 3 describes temporally the stages of implementation of the proposed functioning of the autonomic element during the unsupervised phase. Initially, at time t_0, the autonomic element has only the model for normal traffic. In between times t_0 and t_1, the proposed autonomic element is performed in the presence of normal flows and classes of attacks. During this period abnormal sub-flows are stored in short-term memory of unkown. At time t_1, temporary memory is filled and moves the analysis of sub-flows into candidate clusters, validating and classifying them into innovations and changes in concept. From the time t_1, the autonomic element has proposed the concept of related clusetrs and normal clusters validated and classified as concept drift and novelty.

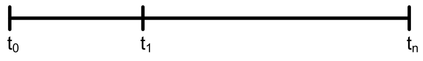


Fig. 3. Incorporation of novelty.

IV. Data and Measurements

Considering the difficulties to obtain actual samples of malicious traffic properly identified, we used artificially generated attacks and inserted into traces that are supposedly free of attacks to validate our experimental autonomic element. Thus, it is possible to verify whether the attacks can be correctly detected by our autonomic element.

To represent the normal traffic were used samples collected from a network gateway at the University of Fortaleza, during the period April 26-28, 2010. The captured packets were temporarily stored and the flows were reconstructed. Each flow was labeled with an application class. The process of labeling each flow was performed in a semiautomated manner through the use of the payload inspection tool OpenDPI [7]. Aiming to ensure that normal traffic really was free of malicious traffic, was also applied to each flows the tool Snort NIDS [8], with rules dated from November, 2010.

Our autonomic element was trained and validated through the use of 3 traffic datasetsts (referred as T1, T2 and T3). Each dataset was collected during periods of 1 hour (morning, afternoon and evening) and they contain the network traffic of the following classes: HTTP(browsers) and Snort Alerts (Alerts of possible flows of malicious activities identified by Snort [8]). Also were added to each dataset, 1250 attacks flows: Denial of Service and Brute Force, artificially generated.

The process of generating of the attack traffic, was conducted in a controlled manner in the laboratory. Figure 4 shows the topology used to generate the attack traffic.

In this scenario, we used three computers: the attacker, the victim and sniffer. The attacker has the role of sending traffic from attacks against services like FTP, SMTP and HTTP, to a victim. To generation the Denial of Service Attack and Brute Force Attack, were used the tools Heyenae [9] and Brutus [10], respectively.

Table II describes the classes, applications and total flows found within each dataset.

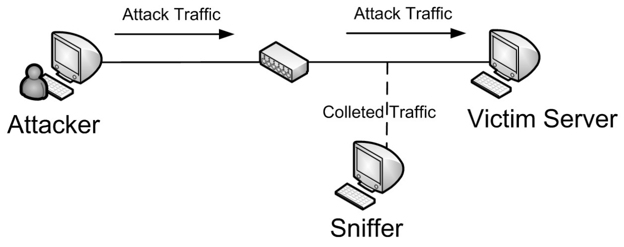


Fig. 4. Network topology used in the generation of artificial attacks.

V. Results and Discussion

In this section, we present the experimental results obtained with the implementation of our autonomic element using the process of cross-validation with 10 partitions [11]. To measure the accuracy of our autonomic element, we use the average rate of false-positive error e_{FPos}, which measures the average error committed when examples of sub-flows of attacks are considered as normal and average of false-negative error e_{FNeg}, which measures the average error committed when normal traffic flows are labeled as attacks. These metrics are given below:

$$e_{FPos} = \frac{\textit{Total of false-positive sub-flows}}{\textit{Total of normal sub-flows}} \quad (6)$$

$$e_{FNeg} = \frac{\textit{Total of false-negative sub-flows}}{\textit{Total of attacks sub-flows}} \quad (7)$$

In the supervised phase in order to determine the set of statistical variables that best characterizes the traffic to be considered normal, we used 50% of HTTP flows and 50% of Snort Alert flows of T1, T2 and T3. The remaining HTTP flows and Snort Alert, along with attack flows of T1, T2 and T3, were used in the unsupervised phase, to verify the learning ability of the our autonomic element.

TABLE II
Summary of Datasets.

Class	Number of Flows		
	T1	T2	T3
HTTP	6835	5284	2588
Snort Alerts	1845	1296	697
Denial of Service	1250	1250	1250
Brute Force	1250	1250	1250

A. Supervised phase

When considering the use of discriminating variables, it is essential to have measured in the sample elements, variables that can really distinguish the population, otherwise the quality of the classification will be compromised. A very common mistake is to think that increasing the number of discriminators, a better solution is reached. The Java implementation of the Wrapper [12] evaluator found in Weka [13] was used for selection of features that better characterize the flows associated to normal concept. Wrapper evaluates features using precision estimations produced by the learning algorithm that will be used on the classification. In this case, the Naïve Bayes and Best First were selected as search method for the Wrapper evaluator.

Figures 5 and 6 show, respectively, the results of the rates of e_{FPos} and the HTTP flows considered unknown in datasets T1, T2 and T3 for the set features selected by Wrapper for N initial packets.

As we can observe on Figures 5 and 6, the initial 11 packets utilization of each flow, was the result with lower average rates of e_{FPos} unknown HTTP flows considered by normal concept generated and so used by the autonomic element in the unsupervised phase.

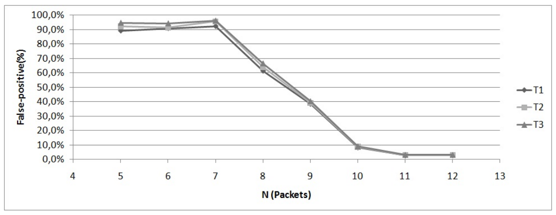


Fig. 5. Rate of false-positive error.

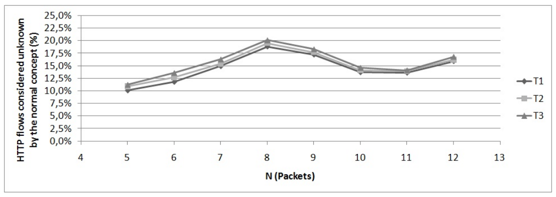


Fig. 6. HTTP flows considered unknown by the normal concept.

B. Unsupervised phase

The tabulated data below represent the results found by applying our autonomic element over 50% remainder set of HTTP flows 50% of Snort Alert flows and Denial of Service and Brute Force flows.

TABLE III
PERCENTAGE OF FLOWS CONSIDERED NORMAL.

Dataset	Class			
	HTTP	Snort Alert	Denial of Service	Brute Force
T1	66,11%	4,03%	0%	0%
T2	69,11%	3,12%	0%	0%
T3	67,51%	3,87%	0%	0%

Table III presents the percentage of flows considered normal by the autonomic element. We can see that on average 67.59% of HTTP traffic was correctly classified as normal, ie, were explained by the model correctly formed during normal supervision. We also had an average of 3.67% of false positive, flows of Snort Alert class that were classified as normal trafic. No Denial of Service and Brute Force attack, were classified as normal traffic flows.

TABLE IV
PERCENTAGE OF FLOWS CONSIDERED AS UNKNOWN.

Dataset	Class			
	HTTP	Snort Alert	Denial of Service	Brute Force
T1	33,89%	95,97%	100%	100%
T2	30,86%	96,88%	100%	100%
T3	32,49%	96,13%	100%	100%

Table IV presents the percentage of flows considered as unknown by the autonomic element. We can see that on average 32.15% of HTTP flows were classified as unknown. It is noteworthy that the initial error rate may indicate a slight concept drift of such flows. We also see that our autonomic element, obtained an average accuracy of 96.33% of Snort Alert flows and 100% in classes of attacks, which implies a high degree of discriminantion of the model generated during the supervised phase.

Tables V and VI show the Concept Drift and Novelty percentage of candidates clusters generated by autonomic element after analysis on the short-term memory.

TABLE V
CONCEPT DRIFT PERCENTAGE OF CANDIDATES CLUSTERS.

Dataset	Class			
	HTTP	Snort Alert	Denial of Service	Brute Force
T1	100%	0,83%	0%	0%
T2	100%	0,78%	0%	0%
T3	100%	0,81%	0%	0%

Table V presents the percentage of candidates clusters endorsed by our autonomic element of the type concept drift. We can observe that from the flows HTTP classified as unknown, 100% of there are classied concept drift, which indicates a slight concept drift of these flows relative to the normal concept determined during supervised phase. We also see that we had an average rate of 0.81% of Snort Alert flows regarded as the type concept drift, which may explain the rate of 3.67% of false positive found by our autonomic element. Table V also shows that none of the flows of the classes of attacks Denial of Service and Brute Force, generates clusters of the type concept drift.

TABLE VI
NOVELTY PERCENTAGE OF CANDIDATES CLUSTERS.

Dataset	Class			
	HTTP	Snort Alert	Denial of Service	Brute Force
T1	0%	99,17%	100%	100%
T2	0%	99,22%	100%	100%
T3	0%	99,19%	100%	100%

Table VI presents the percentage of candidates clusters endorsed by our autonomic element considered novelty. We can observe that none of the HTTP flows generated candidate clusters like novelty, which indicates a high degree of discriminator the of chosen variables and the normal model, both made during supervised phase. We can also observe that our autonomic element found a rate of 99.19% of candidates clusters of the Snort Alert class as type Novelty and 100% accuracy in the classes of attack artificially generated, showing once again the high discriminat degree of the variables chosen and the quality of the model generated during normal supervision.

From the results presented, we can observe a high capacity of the our autonomic element to identify the traffic considered normal, with an average accuracy of 98.77%. It noteworthy also that all normal sub-flows considered unknown, generated concept drift clusters and all sub-flows of the classes of attacks, were correctly classified as unknown and generated novelty clusters.

VI. Conclusions and Future Works

In this paper, we presented an autonomic element to provide self-protection and self-configuration in a computer network, that makes use of OLINDDA, a novelty and concept drift algorithm in data streams. The OLINDDA uses a cluster-based approach, which considers the problem of detecting concepts from an one-class classification perspective.

The implementation of the proposed autonomic element occurs in two phases: the first is called supervised phase and the second phase called unsupervised. In the first phase, the autonomic element determines the statistical variables that best discriminate the traffic considered normal(attack free) from a trace previously classified. In the unsupervised phase, our autonomic element implementation executes the autonomic cycle, using the set of variables and the normal concept is determined during the supervised phase.

From the results presented, we can observe that the implementation of our autonomic element showed an average accuracy of 98.77% of all flows classified as malicious, which indicates a high degree of accuracy in the classification of flows and a strong learning ability of our autonomic element. Despite our autonomic element has been applied in a small set of attacks, it can be perfectly applicable to other types of attacks. We intend in future work, add new attacks and simulate more realistic scenarios.

References

[1] T. R. M. Braga, F. A. Silva, L. B. Ruiz, and H. P. Assunção, "Redes autonômicas. anais dos minicursos do 26 simpósio brasileiro de redes de computadores," *SBC*, pp. 159–208, 2006.

[2] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," in *IEEE Computer Society*. 36(1): 41-50, 2003.

[3] S. A. Mingoti, *Análise de Dados Através de Métodos de Estatística Multivariada: Uma Abordagem Aplicada*. Editora UFMG, 2007.

[4] E. J. Spinosa, A. P. de Leon F. de Carvalho, and a. Gama, Jo "Olindda: a cluster-based approach for detecting novelty and concept drift in data streams," in *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2007, pp. 448–452.

[5] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, pp. 129–137, 1982.

[6] T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world ip networks," *Local Computer Networks, Annual IEEE Conference on*, vol. 0, pp. 369–376, 2006.

[7] "Ipoque's dpi software's open source version." http://www.opendpi.org (as of November, 2010), 2010.

[8] "Snort is an open source network intrusion prevention and detection system (ids/ips)." http://www.snort.org (as of November, 2010), 2009.

[9] Hyenae, "Hyenae project (hyenae)." http://hyenae.sourceforge.net (as of November, 2010), 2010.

[10] Brutus, "Brutus - the remote password cracker." http://www.hoobie.net/brutus (as of November, 2010), 2001.

[11] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Pearson Education, 2003.

[12] M. A. Hall, "Correlation-based feature selection for discrete and numeric class machine learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 359–366. [Online]. Available: http://portal.acm.org/citation.cfm?id=645529.657793

[13] E. F. Ian H. Witten, *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2005.

Dynamic Model-based Management of a Service-Oriented Infrastructure

Félix Cuadrado, Rodrigo García-Carmona, Juan C. Dueñas
Departamento de Ingeniería de Sistemas Telemáticos
ETSI Telecomunicación – Universidad Politécnica de Madrid
Madrid, Spain
{fcuadrado, rodrigo, jcduenas}@dit.upm.es

***Abstract*- Models are an effective tool for systems and software design. They allow software architects to abstract from the non-relevant details. Those qualities are also useful for the technical management of networks, systems and software, such as those that compose service oriented architectures. Models can provide a set of well-defined abstractions over the distributed heterogeneous service infrastructure that enable its automated management. We propose to use the managed system as a source of dynamically generated runtime models, and decompose management processes into a composition of model transformations. We have created an autonomic service deployment and configuration architecture that obtains, analyzes, and transforms system models to apply the required actions, while being oblivious to the low-level details. An instrumentation layer automatically builds these models and interprets the planned management actions to the system. We illustrate these concepts with a distributed service update operation.**

Keywords- Model-Based Management; Runtime Models; Service-Oriented Architecture; Service Configuration; Service Deployment.

I. Introduction

In engineering, models are abstractions or conceptual objects used in the creation of a system. Model Driven Engineering is a methodology based on the systematic use of models in software development processes [1]. It aims to alleviate the complexity of current IT applications and infrastructure platforms and express domain concepts effectively. In their usage as software and hardware development tools, models become blueprints to build systems. The initial models are increasingly refined and enhanced by means of transformations, until the final system (the code) emerges [2]. The final model should represent the system "as it must be". Reality shows us that this source code must be frequently refined or modified by hand: code is changed throughout its lifetime and it is seldom synchronized with the original design models. In systems engineering the same situation happens; network and system design models, once defined and deployed, are disconnected from the real situation. This problem is exacerbated after release, during their operation time. Each change further decouples the models from the reality, invalidating the model for runtime reasoning.

Because of these limitations, we propose to use dynamic models; whose grammar (the metamodel) is defined beforehand, during the design phase, but whose information or data is obtained in runtime. These models are a key tool for system management, as they allow reasoning over the system "as it is", and conform with the system "as it must be". Thus, it is possible to automate the decision making activities related to system management, and to perform them based on these models depicting the actual state. We have combined static and dynamic models of running systems, in order to provide automated deployment and configuration for service oriented architectures.

The article is structured as follows. Next section provides an overview on the context of application, identifying the main requirements and concerns. Section 3 first discusses the suitability of the main existing information model standards for autonomic service management, and after that presents our proposed information model abstractions. The next section presents a management architecture that builds on the dynamic model approach. Section 5 expands the main ideas of our proposal through the explanation of a case study. Finally, the article is closed with the main conclusions and potential lines of future work derived from the presented results.

II. Context of Application

The increased importance of IT infrastructure has led to significant investments in infrastructure, which must be amortized over long periods of time. However, systems evolve rapidly, rendering purchased units as legacy technology before their lifetime has been completed. On top of that, it is necessary to upgrade applications and enterprise services, and acquire new equipment, in order to continuously improve process efficiency to gain a competitive advantage. Thus, systems are composed by not only legacy systems, mainframes, or databases, but also Java Enterprise Edition (JEE) application servers, or Business Rule Manager (BRM) systems. The resulting enterprise infrastructure is a complex heterogeneous distributed system, composed by dozens of different servers and application containers, deployed over hardware machines interconnected through complex network distributions, containing firewalls, virtual private networks and other access restriction and security mechanisms.

Functional interoperability between all the components of the IT infrastructure is usually achieved by adopting a higher-level integration layer, which is based on Service Oriented Architecture and Business Process Management

(SOA/BPM). This way, each artifact of the system is presented as a service, hiding its implementation details and providing a uniform high-level view. Services are published in directories and connected through an Enterprise Service Bus (ESB), where additional non-functional capabilities can be added to the communications, like logging, or data transformation. On top of that, BPM technologies, such as BPEL (Business Process Execution Language) engines, orchestrate the activities, bridging the gap between the IT infrastructure and the business processes.

The SOA/BPM abstraction maximizes the use of the existing IT infrastructure, but managers still have to cope with the underlying heterogeneity and complexity, while supporting three key business requirements: controlling operation costs, warranting the quality of services and handling the evolution of the services and the infrastructure.

Traditional management processes are identified with human operation over a management administration console. Monitoring information and events are collected and aggregated into the console, and the administrator invokes specific operations on the environment based on the identified objectives and the collected information. Operations are executed in scripts, containing the exact set of machine-specific instructions for achieving a specific task. Because of that, scripts lack reusability and suffer from the increased complexity, distribution and heterogeneity of current IT systems.

The limitations of this approach become more evident as the complexity and heterogeneity of the managed systems keep growing. Changes to the environment impact the complete management process, as the configuration and workflows must be manually adapted to the specifics of the environment. Management operations are manually created and composed, requiring specialized knowledge from the administration experts, and can hardly be reused. On top of that, a runtime system configuration has dependencies between heterogeneous artifacts, propagating the impact of any change or error throughout the whole system.

There is clearly a necessity of reducing complexity, and lessening human intervention by automating parts of the management processes. These problems can be alleviated by using models; Model Driven Management [3] is a new approach for management, where models allow the abstraction from the complexity of the environment. This approach has numerous advantages over others, thanks to the greater expressivity of models [4]. We build on this approach, and propose the need to handle both static models containing the definitions of the developed services, and dynamic models that contain information directly obtained from the running systems (monitoring information); these models must be related and transformed in order to generate control actions that will be themselves represented by models, able to be applied on the managed systems by the proper agents. To verify this approach we have built a deployment and configuration system, which operates on models for characterizing the services, the runtime environment and the operations on a service oriented architecture.

III. Management Information Model

The management of networked systems is defined by [5] as all the measures necessary to ensure the efficient and effective operation of a system and its resources, based on the organization goals. However, the scope of distributed management greatly varies ranging from network, resource, application, service and business concepts. Regardless of the scope, every management system requires an information model that provides a homogeneous view of the managed elements. While there are successful proofs of concept of the implementation of autonomic managers using ad-hoc models and ontologies [6], the lack of alignment to existing standards and models greatly complicates its applicability to general cases. The information model must include all the relevant information for the management operations, the elements, its characteristics and relationships, while at the same time it must be flexible enough to adapt to heterogeneous environments and be as compatible as possible with the existing information modeling standards.

A. Information Modeling Standards

Several standards have been defined for modeling the relevant management information of a distributed system. Alternatives range from mature standards from the network management domain to emerging initiatives from the Internet domain. Here we present a brief overview on the most relevant ones, showing how they support service management.

MOWS (Management Of Web Services) [7] is an OASIS standard that defines how to represent Web Services as manageable resources. MOWS is part of the Web Services Distributed Management (WSDM), a set of standards from OASIS devoted to the management of IT distributed systems using Web Services technology.

The Common Information Model (CIM) [8] is an information model standardized by The Distributed Management Task Force (DMTF) industrial association. CIM is an object-oriented model for describing overall management information in a networked enterprise environment. CIM is specified as a set of UML models and complementary MOF (Managed Object Format) files, textual files expanding the semantics of the defined elements. CIM is a modular, extensible standard; it models a very broad set of elements including databases, networks, user preferences, and applications among others. Internally, CIM is divided into a core model, defining the basic elements, and additional models extending from the base elements with additional details on one specific area (application, network, computer are some examples of profiles).

The OMG Deployment and Configuration Model [9] provides a simple and flexible model for representing deployment and configuration operations over a distributed target. The target environment is called a domain in its terminology, and is described by an object-oriented information model. The base elements of the domain model are resources, which are named entities classified into one or more types. Resource instances model physical artifacts, mainly: nodes, bridges and links.

There are two common characteristics of information models that cover systems and services: the use of object-oriented abstractions and the resource concept as the essential unit of management.

B. *Proposed Information Model*

After evaluating the existing information modeling standards, we have defined a set of modeling abstractions that try to effectively capture the relevant information of a heterogeneous distributed environment. The metamodels build upon the common ground shared by the standards, with D&C being the base reference because of its flexible nature.

Our service deployment and configuration architecture is supported by three metamodels, governing the static definitions of services, the runtime description of the environment and the planned management operations to the runtime environment. These metamodels complement each other, so they completely cover the required information for the management of enterprise services. The metamodels have been defined in EMOF (Essential MOF, a subset of the OMG's Management Object Facility MDA language)

The three metamodels share a core concept that is the base of both static and dynamic abstractions; the resources. A resource is a manageable element. Resources are characterized with a name, a version identifier and a set of properties. The resource definition is complemented by a type field, which establishes a resource taxonomy, inherently classifying the basic assets of the infrastructure environment (ranging from services to containers). This allows management systems to define actuators and policies that automatically apply to the matching elements. The concept has been taken from OMG D&C and expanded with versioning information for software resources.

The software metamodel provides a software architect-friendly abstraction for modeling software components, known as deployment units in our terminology. Units model their provided services and external requirements using the resource concept. Finally, deployment units can also specify environment constraints needed for correct performance (such as existing system services, available disk space or minimum amount of RAM memory) as mandatory resources of the runtime domain. Examples of typical deployment units include Java EE WAR and EAR files, Database DDL (Data Definition Language) and DML (Data Modification Language) scripts or BPEL (Business Process Execution Language) process definitions. Instances of this metamodel are considered static for the sake of configuration and deployment, because they are inputs to the management processes coming from the development infrastructure. The software model combines aspects from CIM Application model with the resource concepts. A previous version of this model is presented in [10].

Once components and services are deployed, they become runtime entities that are part of the environment, with a state and a specific configuration. The runtime metamodel defines the topology and the configuration of the managed environment. An environment is composed of a set of distributed nodes, interconnected through a network. Node configuration information is also modeled as resources and properties, whereas the network structure is defined by D&C's interconnect and bridge elements. On top of that, nodes host containers, where components and services are deployed. Examples of containers include an application server, a business process manager or a database. Runtime units are the main elements of the metamodel. They represent deployed units, providing status and runtime configuration information. Runtime models are generated on the fly, as they represent the environment for management purposes. Therefore, we need agents to extract the information and populate the models. The combination of both metamodels keeps the traceability between the static view of software development and the dynamic view of service management.

Finally, we have defined the management operations that can be applied over the runtime elements (resources, containers, and deployed units) as the plan metamodel. Because of the dependencies and inter relationships existing in a distributed system, operations cannot be executed individually and must be aggregated in plans. A plan is a collection of activities (such as unit deployment, component activation, or resource configuration) which must be executed over the environment to achieve a management objective (e.g., release a new version of the client user application). The activities are included in a directed graph, ensuring a correct execution order while allowing at the same time parallelization of non-dependant tasks. This definition constitutes a management DSL (Domain Specific language). Activities refer to the concepts described in the software and runtime metamodels (such as deployment units, resources, configuration values and containers). This abstraction allows a management system to dynamically create plans for achieving a specific goal through model reasoning, instead of the traditional mechanism of a system administrator manually defining change workflows.

IV. SERVICE MANAGEMENT ARCHITECTURE

In an enterprise environment the availability of the complete runtime information is vital for an effective management. The impact of changes spreads over the environment, due to the nature of heavily distributed systems. Additionally, the heterogeneity of the elements complicates coordinated efforts [11]. Because of those factors, our management functions are centralized for the complete environment, and reason over generic models abstracting from the specific system details.

Fig. 1 provides a high-level overview of our system. On the left-hand side we can see the main blocks of the management system (such as plan generation, Service-Level Agreement monitoring, or environment asset management). The management functions provided by the architecture range from static information processing (e.g., unit description storage, configuration and policy definition) to dynamic reasoning (e.g., environment monitoring, and plan model generation). The inputs and outputs of these components are model instances of the previously described metamodels, which isolates the functional components from the specifics of the managed environment. Additionally, the architecture implements a closed control loop. The

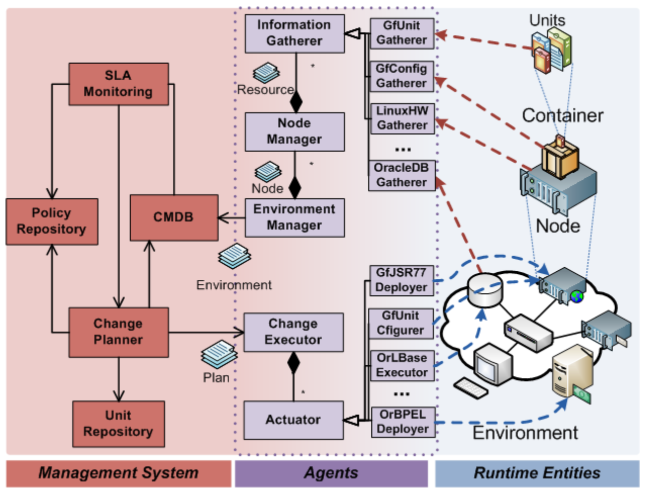


Figure 1 High-level Architecture View

configuration and status are continuously monitored by the agents, and transformed into runtime models. This information is processed by the systems, also taking into account high-level objectives and policies. The results of these processes are the deployment and configuration change plans, which are interpreted by the agents.

The physical managed elements are pictured on the right hand side. The impedance mismatch is addressed by the agent layer (in the central part of the figure), which transparently instruments the runtime elements and builds a coherent model with all that information. Communication goes both ways. The agents produce a model abstraction of the physical environment for the management blocks. At the same time, they apply plan models to the runtime infrastructure.

Depending on enterprise policies, business requirements and architecture decisions the composition of a runtime environment differs significantly. The instrumentation layer must automatically adapt to different physical configurations, without the need of manually defining the topology of the specific environment. Because of that, the agents' architecture follows a layered, extensible model.

The top-level agent is the *Environment Manager*, which acts as the contact point between the environment and the management system. It provides model descriptions of the environment and orchestrates the execution of plan models. Internally, it uses the DNS-SD protocol for automatic agent discovery. This way, the environment topology is dynamically built, although it is also possible to manually populate it to reflect runtime systems behind restrictive firewalls. The discovered agents are the *Node Managers*, which govern the resources, configuration and services available at node level. Information about hardware and software resources, containers and application is captured to the environment model by the Information Gatherer agents. These components instrument a specific aspect of the node, such as hardware information, JEE application server resources or service configuration. A *Node Manager* collects the gathered models for providing to the *Environment Manager* a complete characterization of the node. The operation infrastructure is composed of the *Change Executor*, which receives execution orders for change plans to the environment, and multiple *Actuators*, which apply the activities to the physical elements. These elements are aggregated similarly to *Gatherers*; each one is capable of applying one or more operations (i.e., configure container, install deployment units) on some parts of the environment. We can see how the base instrumentation elements are generic, and only the endpoint *Gatherers* and *Actuators* mediate between the runtime models and the runtime system. These agents perform the key transformations for dynamic model management, as they convert Specific Models from each management interface to our Platform Independent Model (the runtime metamodel), and interpret our plan DSL elements as invocations to vendor-specific commands.

V. CASE STUDY

We will present an industrial case study to illustrate how the elements of the management system collaborate for obtaining and applying configuration and deployment changes. The scenario has been extracted from the ITECBAN project, a Spanish Research project from the CENIT program whose objective is to develop a SOA-based core banking solution. A banking organization internally uses a credit grant service. It combines the use of inference engines and the input of human experts to provide a response to the requestor. On a technical view, the service is composed of clustered JEE applications, BPEL processes, Business Rule definitions and database information. These components are deployed over a distributed environment, and communicate through Web Services. After a user reports a service fault, the incidence is escalated to the development team, which releases an updated version of the faulty service, and a change request is issued to provision these modifications to the runtime environment.

Service update is a complex process, involving installation, life-cycle control and configuration of several, inter-dependent artifacts. Its correct execution requires retrieving and processing information about the current environment state, and the logical changes to the affected services. We will describe how this scenario is supported by collecting the state from the physical elements, obtaining a change plan from the static and dynamic models, and applying the changes to the environment.

The initial step is to obtain an updated snapshot of the environment, which will be represented by an updated runtime model. At this point, each *Information Gatherer* connects to the management interfaces of a monitored part of the infrastructure, obtains its current state and transforms that information to our modeling abstractions. As an example, Glassfish (the reference implementation of the JEE standard) *Gatherers* access the remote JMX Server, query and retrieve the relevant JSR 77 MBeans and transform that information into container, unit, resource and property elements. An analog process is applied to the rest of instrumented infrastructure, such as Oracle configuration and information, and Hyperic HQ inventory model instances. The *Node Managers* and *Environment Manager* aggregate that

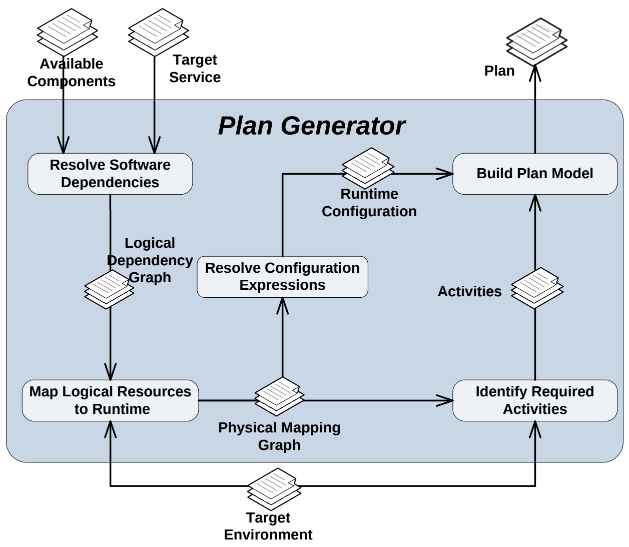


Figure 2 Plan Generation Activity Diagram

information and provide an updated environment model to the management system.

Once the updated model has been generated, the management components analyze the objectives and current status, and produce a change plan that will contain the required changes for updating the selected service. Fig. 2 shows the general update change execution flow. Functional elements operate with model instances, both as input parameters and as execution results.

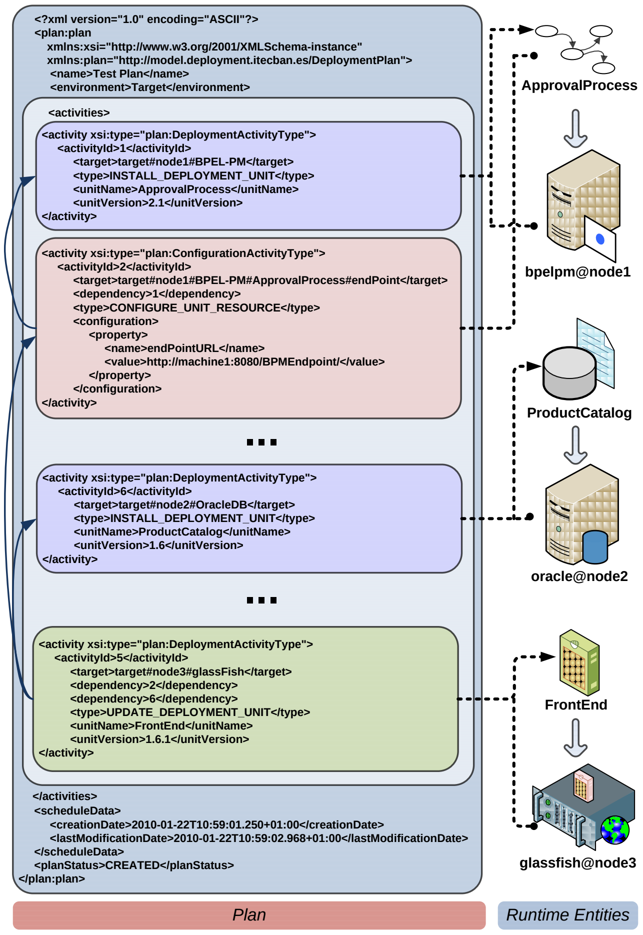
The process starts at the *Resolve Artifact Dependencies* step. The plan generator takes the descriptor of the service to be updated, along with the models of all the available components, and obtains a model graph representing the logical dependencies of the provider. Both current and planned versions are analyzed, in order to estimate the impact of the operation to the rest of the system. In case the update operation can be safely executed, the workflow goes on in order to generate the update plan.

After the logical dependencies have been calculated, the plan generator retrieves the updated environment model, and combines it with the logical graph in order to *Map the Logical Resources to Runtime elements*, designating a container for each logical unit (deployable artefact). Whenever more than one container is suitable for a given component, a decision is made either by an administrator or a distribution policy.

After obtaining the physical component distribution, the process will *Identify the Required Activities* for reaching the desired state, as well as the restrictions in their execution order. The state of the runtime environment is also taken into account in order to only generate the mandatory activities, avoiding redundant actions (e.g., if a deployment unit is already running on the selected container, an installation activity won't be generated for it). In parallel to activity generation, the executor will *Resolve the Configuration Expressions* of the graph components, automatically calculating the required configuration changes to services and environment in order to work correctly after the update operation.

Finally, by combining the configuration parameters and the plan activities, the change executor *Builds the Plan Model*. The complete plan is composed by 36 activities that represent the management operations. Each activity defines the operation to be performed, the element of the environment (such as a container resource, or a runtime unit) where it will be applied, and the operation arguments. Fig. 3 shows a fragment of the resulting plan, composed by four activities represented in XML syntax: two installation activities, one update activity and one configuration activity. The plan also contains the mandatory dependencies between them, to assure its correct execution.

Once the plan has been created, it is processed by the *Change Executor*. First, it identifies which *Actuator* will perform each activity. The matching between plan activities and *Actuators* is made using the resource type taxonomy previously mentioned. *Actuators* interpret the generic activities defined by the model into commands of the languages supported by the specific management interfaces. As an example, activity #5 (update a WAR artifact deployed at the Glassfish node3 cluster) is translated into the "DeploymentManager.redeploy" operation, which is defined



```
<?xml version="1.0" encoding="ASCII"?>
<plan:plan
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:plan="http://model.deployment.itecban.es/DeploymentPlan">
    <name>Test Plan</name>
    <environment>Target</environment>
    <activities>
<activity xsi:type="plan:DeploymentActivityType">
    <activityId>1</activityId>
        <target>target#node1#BPEL-PM</target>
        <type>INSTALL_DEPLOYMENT_UNIT</type>
        <unitName>ApprovalProcess</unitName>
        <unitVersion>2.1</unitVersion>
</activity>
<activity xsi:type="plan:ConfigurationActivityType">
    <activityId>2</activityId>
        <target>target#node1#BPEL-PM#ApprovalProcess#endPoint</target>
        <dependency>1</dependency>
        <type>CONFIGURE_UNIT_RESOURCE</type>
        <configuration>
            <property>
                <name>endPointURL</name>
                <value>http://machine1:8080/BPMEndpoint/</value>
            </property>
        </configuration>
</activity>
...
<activity xsi:type="plan:DeploymentActivityType">
    <activityId>6</activityId>
        <target>target#node2#OracleDB</target>
        <type>INSTALL_DEPLOYMENT_UNIT</type>
        <unitName>ProductCatalog</unitName>
        <unitVersion>1.6</unitVersion>
</activity>
...
<activity xsi:type="plan:DeploymentActivityType">
    <activityId>5</activityId>
        <target>target#node3#glassFish</target>
        <dependency>2</dependency>
        <dependency>6</dependency>
        <type>UPDATE_DEPLOYMENT_UNIT</type>
        <unitName>FrontEnd</unitName>
        <unitVersion>1.6.1</unitVersion>
</activity>
   </activities>
   <scheduleData>
     <creationDate>2010-01-22T10:59:01.250+01:00</creationDate>
     <lastModificationDate>2010-01-22T10:59:02.968+01:00</lastModificationDate>
   </scheduleData>
   <planStatus>CREATED</planStatus>
</plan:plan>
```

Figure 3 Generated Deployment and configuration plan Model

by the JSR 88 Deployment Management API offered by the Glassfish server. Finally, the *Change Executor* invokes the *Actuators* in the order dictated by plan dependencies, and as a result the runtime environment configuration is updated to support the new version of the credit grant service. Additional details on the instrumentation architecture are presented in [12].

VI. CONCLUSIONS

In this paper we have introduced the usage of dynamically generated models for enabling autonomic management of service oriented architectures. The proposed abstractions define both static models for descriptions of software components before they are deployed to the runtime environment, and dynamically generated models (obtained by monitoring agents) for the runtime environment elements. By mixing, transforming, aggregating and processing these models, we obtain a third model, the deployment and configuration plan, which is generated and consumed at runtime. Controlling agents are in charge of interpreting and executing it so the environment is changed accordingly to the plan.

Our interpretation of MDM offers several improvements to the state of the art: there are clear, unambiguous but extensible metamodels for all the information handled; static relations between models are represented at the metamodel stage, while relations between dynamic models are ensured by the proper transformations; transformations can take several models to produce a new one (as expressed in); agents are considered as producers or consumers of models, thus behaving as the frontiers of the management system; the runtime system can be observed at a high level of abstraction, at its logical view, but at the same time its model represents a real situation, the system "as it is".

Once the models have reached a certain degree of maturity, and we have implemented a proof of concept of the system, we consider the whole set composed by metamodels, transformations, implementation of management functions and monitoring and control agents, as a complete infrastructure for the management of service oriented architectures. After successive versions of the system we have provided a method for the development of management functions, based on the provision of transformation on models, and the generation, adaptation or usage of the required agents for monitoring and control.

We are currently adding more management functions that will cover the full range of activities related to services lifecycle. In its industrial application it is also necessary to provide enhancements such as security and access control, persistence and management of the intermediate and final models, instrumentation for virtualized systems, generation of reports for business intelligence and integration with IT service level frameworks.

ACKNOWLEDGMENT

The work presented in this article has been partially funded by the ITEA OSAMI-Commons and CENIT ITECBAN R&D projects. The authors are grateful to MITYC (Ministerio de Industria, Turismo y Comercio) and CDTI (Centro para el Desarrollo Tecnológico e Industrial) for its support.

REFERENCES

[1] D. Schmidt, "Model-Driven Engineering", IEEE Computer, vol. 39, no. 2, February, 2006.

[2] L.A. Fernandes, B.H. Neto, V. Fagundes, G. Zimbrao, J.M. de Souza, and R. Salvador, "Model-Driven Architecture Approach for Data Warehouse", Proceedings of the 6th International Conference on Autonomic and Autonomous Systems, pp. 156-161, Cancun, Mexico, 2010.

[3] M. Barbero, F. Jouault, and J. Bézivin, "Model Driven Management of Complex Systems: Implementing the Macroscope's Vision", Proceedings of the 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, IEEE Computer Society, 2008.

[4] V. Talwar, D. Milojicic, W. Qinyi, C. Pu, W. Yan, and G. Jung. "Approaches for service deployment" IEEE Internet Computing Magazine, vol. 9 Issue 2, pp.70-80, 2005.

[5] Hegering, H., Abeck, S., Neumair, B, "Integrated Management of Networked Systems. Concepts, Architectures and Their Operational Applications", Morgan Kaufmann publishers, ISBN: 3-932588-16-9, 1999

[6] J.M. Gonzalez, J.A. Lozano, and A. Castro, "Autonomic System Administration. A Testbed on Autonomics", Proceedings of the 5th International Conference on Autonomic and Autonomous Systems, pp. 117-122, Valencia, Spain, 2009.

[7] K. Wilson and I. Sedukhin, "Web Services Distributed Management: Management Of Web Services (MOWS 1.1)". OASIS, Standard 2006.

[8] DMTF (Distributed Management Task Force) Common Information Model (CIM) specification v2.19. . 2008

[9] Object Management Group. Deployment and Configuration of Distributed Component-based Applications Specification. Version 4.0. April 2006.

[10] J.L. Ruiz, J.C. Dueñas, and F. Cuadrado, "Model-based context-aware deployment of distributed systems" Communications Magazine, IEEE , vol.47, no.6, pp.164-171, June 2009

[11] J. Strassner, "Handbook of network and service administration" Ed. Elsevier, 2007, ISBN 978-0-444-52198-9.

[12] F. Cuadrado, R. Garcia-Carmona, A. Navas and J.C. Dueñas, "A Change Execution System for Enterprise Services with Compensation Support", Conference on Enterprise Information Systems,. Madeira, Portugal. Communications and Computer Series. Vol. 109 pp.441-450. 2010

.

A Dynamic Service Composition using Semantic Ratiocination on Integrated Home Network System

Junsoo Kim*1 Junya Nakata†2 Takashi Okada*3 Marios Sioutis*4 Azman Osman Lim*5 Yasuo Tan*6
* *School of Information Science, Japan Advanced Institute of Science and Technology*
1-1 Asahidai, Nomi, Ishikawa Japan
1*junsoo@jaist.ac.jp* 3*tk-okada@jaist.ac.jp* 4*s091003@jaist.ac.jp* 5*aolim@jaist.ac.jp* 6*ytan@jaist.ac.jp*
† *Hokuriku Research Center, National Institute of Information and Communications Technology*
2-12 Asahidai, Nomi, Ishikawa Japan
2*jnakata@starbed.go.jp*

***Abstract*—Recently, the ubiquitous technologies allow general household appliances to be connected within the network at home. Since a number of researcher have tried to provide these environment as their own field such as UPnP, DLNA, OSGi and ECHONET, The home network systems (hereafter referred to as HNS) are comprised of a networked appliances to provide various services and applications for end users. Therefore, HNS has been realized with an integration of features of multiple appliances. However, to integrate services still has several challenges that a providing of a composed service considering of a policy influences to composite services because an element of the composed service are dynamically changed by priority of the policy. In this paper, we proposed the dynamic service composition using semantic ratiocination on integrated home network system. Our research provided the solution to semantically decide a service element, which is dynamically changed according to a service objective with considering the priority on the ground of the policy. Moreover, our system is able to include an external ontology model by importing its name space. Such distributed development of a service model will improve scalability of the service model.**

***Keywords*-Service Composition; Semantic Ratiocination; Home Network Service.**

I. Introduction

For more than a decade, ubiquitous/pervasive technologies allow general household appliances to be connected within the network at home[1]. Since a number of researcher have tried to provide these environment as their own field such as Universal Plug and Play (UPnP) [3][2], Digital Living Network Alliance (DLNA) [4], Open Services Gateway Initiative (OSGi) [5] and Energy Conservation and Home care Network (ECHONET) [6]. HNS is comprised of such networked appliances to provide various services and applications for home users. Therefore, HNS has been realized with an integration of features of multiple appliances. Especially, providing integrated services to end-user with briefly establishment of the appliances has been considered for actualizing ubiquitous home network environment.

A service (or application) composition method for integrated service in HNS can be divided into two approaches: Proactive (Static) service composition, and Reactive (Dynamic) service composition[7]. For example, ICARIS [8] and eFlow [9] proposed a template based dynamic service composition system that supporting applications involving complex interaction patterns such as conditional branch or iteration which is problem with static service composition.

Since the dynamic service composition does not depend on a human to compose a service, it may have difficulty in composing services. To compose a service dynamically

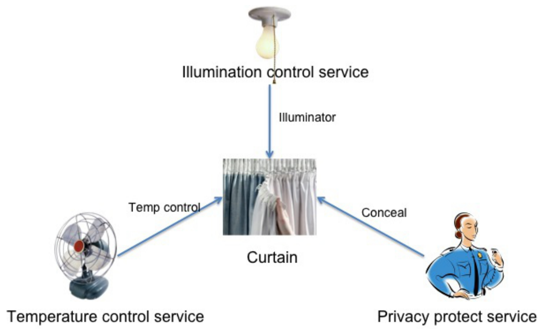


Figure 1. Example of services on semantic considering

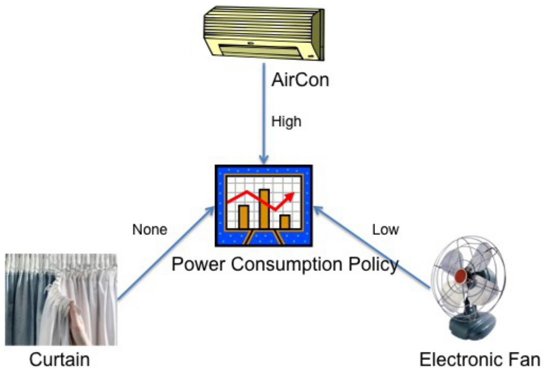


Figure 2. Example of services with policy

without a manipulation of the human, an agent has to understand a meaning of an objective of the service that the agent would provide. In addition, the agent need to recognize what possible service does it has semantically for composing the service.

To solve these problems, Ninja [10] and SeGSeC [11] created an execution path from a user request by performing interface matching based on semantics. These solutions are considered how to matching the interface between services. However, fundamentally, the dynamic service composition in the home network system is needed to perform flexibly depend on home environment.

Imagine the scenarios illustrated in Figure 1. Providing a temperature control service with cooling, when an air conditioner is stopped with trouble, the agent has to decide substitution such as electric fan, window and curtain even these substitution can't reach to target temperature. In this case, the agent has to understand that the curtain can cooling a temperature by excluding a solar heat. As illustrated in Figure 2, If the agent need to consider a policy of the power consumption during provide the cooling service, priority of possible service is changed to the curtain before the air conditioner.

These dynamic service compositions using the semantics are need to resolve number of general issues that the ontology engineering has, suitable definition every notions, representation of extremely diverse relationship between components and optimization of a reasoning engine. Nevertheless, the dynamic service composition has the potential to provide flexible and adaptable services by properly selecting and combining components based on the user request and context.

In this paper, we proposed the dynamic service composition using ratiocination on integrated home network system. Our research provides a flexible service composition in various home environments by making the system to semantically understand an objective of a service. Moreover, it is possible to realize extendable services and priorities, since these semantic descriptions can be built separately in their own area by a name space. In the rest of this paper, Section II gives background of characteristic of a home appliance and brief of proposed idea. Section III discusses our system model about service representation and process annotation. Section IV describes our simple experimental result and Section V gives our conclusions and discusses future works.

II. Background

Recently, functionality of an appliance has been composed by one or more process. Moreover, these processes are possible to used as one by one for other service by defining an interface of each process as showing Figure 3. It means that an agent, which is centralized management system, can use these independent processes to compose (or de-compose)

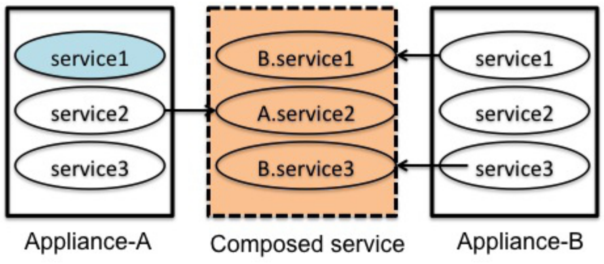


Figure 3. Example of the Service Composition

virtual service. For example, the agent can use temperature sensor that is included in air conditioner for measuring the temperature of a room.

As we mentioned above, the appliance can provide multifunctional service according to an objective of a service. However, to provide service automatically, the agent need to understand a meaning of these service, since the home appliance may have different meaning depending on service objective. For example, the curtain could be included in illumination service, temperature control service and privacy protect service. Another important issue is that a priority of the service is changed by policy. As we consider air condition service with energy consumption, an electric fan or curtain consumes less energy than an air conditioner. However, as considering of capability, the air conditioner may have higher priority than other elements. Hence, the agent needs to understand various policies as many as possible and decide a service according to the policy.

To solve above challenge, we need a knowledge based machine-readable service representation because the agent need to understand a semantics of services and a relation between services. Also, we need to consider a decentralized development setting for defining many of a political matters.

Basically, our system considers that providing a flexible service composition in various home environments as follow.

- Semantical service composition – understanding an objective of service
- Policy based composition – composed element could be changed depend on policies
- Robust to an environment variation – when an element in composed service is stopped, substitution could be found

III. System model

Architecture of our proposed system is shown in Figure 4. First, the agent registers the information of atomic process that can't be divided any more from appliances to the process registration. These atomic processes have an information that is a process description and resource type of input/output. Secondly, when the agent gets request with composing service, the service composition decides what elements will be used based on the semantic representation. Finally, the

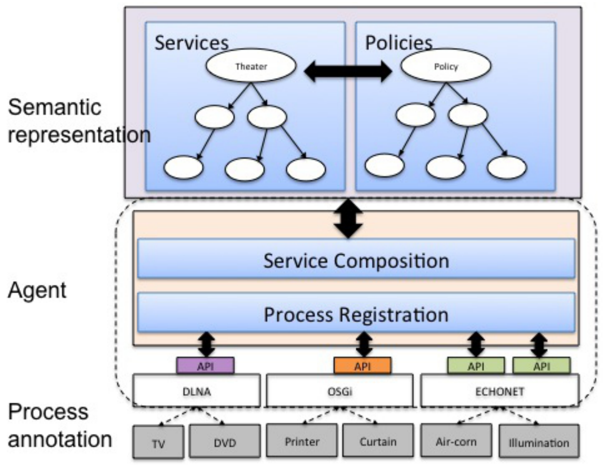


Figure 4. System Architecture

semantic representation defines services semantically using ontology modeling. The service model provides to consider with various policies as importing the policy model. In this section, we discuss specifies of our system. Especially, it'll be focused on representing relations between services and policies.

Figure 5 is that describes the system flow of the service composition module. The service composition module maps a service model with policy by including outside models. When user requests service composition, the service composition module queries with several conditions such as temp, cooling and energy saving. Obtained elements from the mapped service model will be matched with process information from the process register. Finally, process register provides process list to service composition.

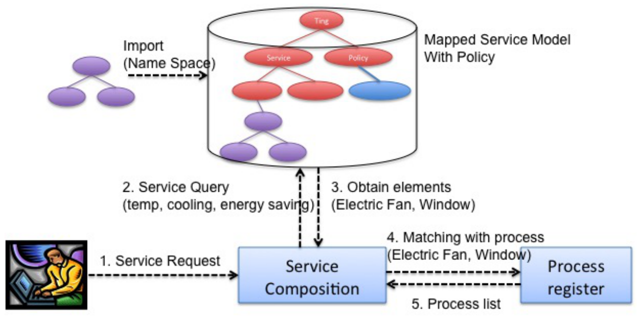


Figure 5. System flow of the Service Composition Module

A. Semantic Service Representation

To define services semantically using ontology modeling, we used a web ontology language (OWL) that is designed for use by applications that need to process the content of information instead of just presenting information to humans[12]. We discuss how to build the service model within a temperature control service as a representative service in this section.

1) Modeling the service structure: As a service structure modeling, it's needed to describe property that service has. For example, temperature control service has a temperature control descriptor as a property. The temperature control descriptor has subclass of functionality of temperature as follow.

- Heating : High, Low, Variable
- Cooling : High, Low, Variable

In addition, the service structure model has information of relation between services. For example, the possible services to use for temperature control are an air conditioner, electric fan and window.

2) Modeling the Services: In the service modeling, the services are represented by their features. the service has property within the service descriptor that is described by the service structure model. For example, each service has property of heating and cooling capability as follow.

- AirConditioner
  - Heating Capability High
  - Cooling Capability High
- Electric fan – Cooling Capability Low
- Window – Cooling Capability Variable

3) Modeling the Policy: As we mentioned before, providing service with considering the policy is influence to composite services because elements of composed service are dynamically changed by priority of the policy. The problem is that only one system could not define much kind of these policies. We believe that ontology will solve the problem by importing a policy, which is already defined. Hence, the policy model provide various point of view to understand services such as following example of power consume policy.

- AirConditioner – High Consumption of Power
- Electric fan – Low Consumption of Power
- Window – None Consumption of Power

B. Services annotation

To annotate a process of appliances, we used an OWL-S (semantic markup for web service) that is an ontology of services that makes these functionalities possible[13]. The OWL-S provides to discover, invoke and composite a web service. However, there is a difference of a characteristic between the web service and home network service as shows follow.

- Appliance has an area of effect or not
- Service has occupancy or not
- Service has depended on capability of a device

When the agent composites a service, service's area of effect has to be defined, since some appliance provides only limited area such as display service or cooling service. These services have another characteristic of the occupancy.

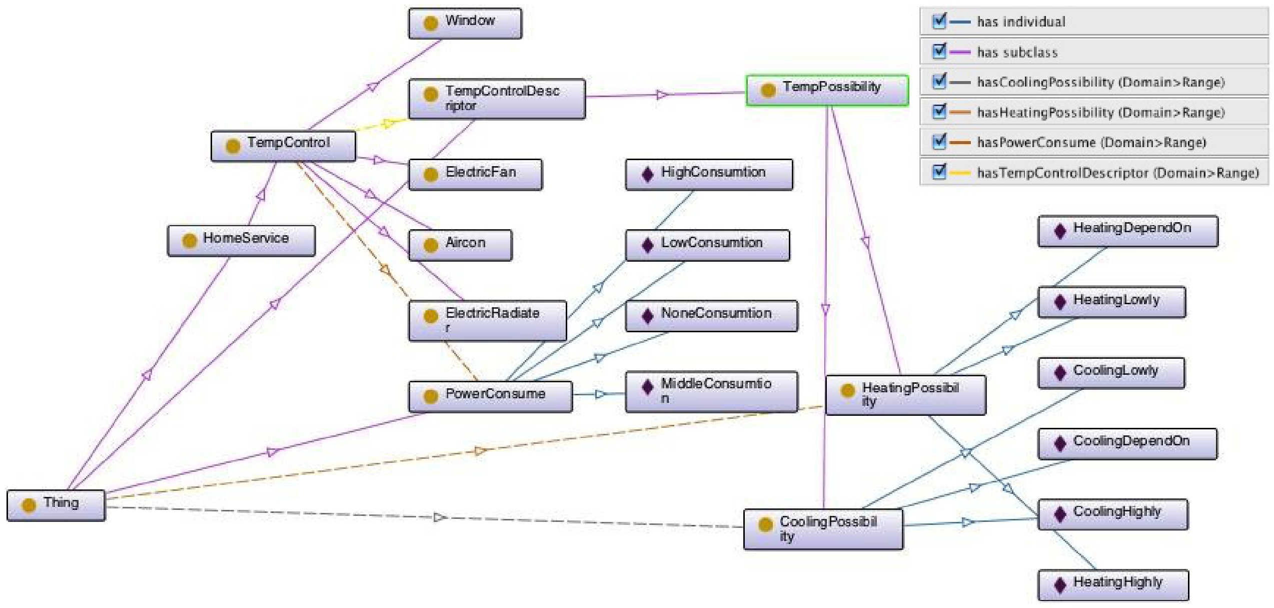


Figure 6. Description of the Serivce Model

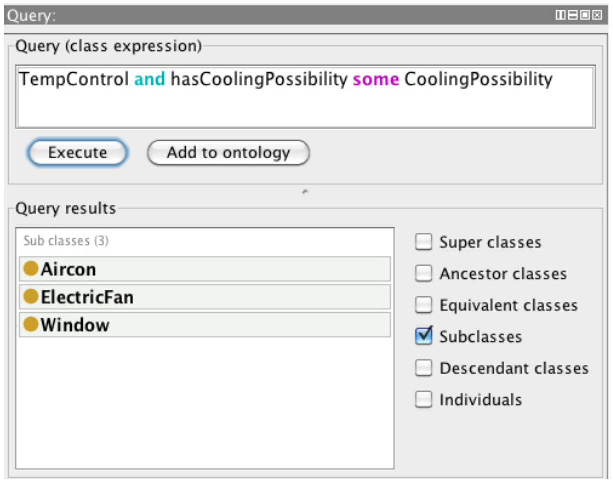
For example, if agent already used display service, other composited service cannot use it. Finally, the appliances have their own capability of process. When the agent decides element among atomic processes that performs as same functionality, the decision will be capability of an appliance.

IV. Experimental result

We have designed and built a simple prototype to demonstrate the reasoning of services elements using semantic service representation. To build an ontology modeling of service and policy, we used the protege that is open-source platform to construct domain models and knowledge-based applications with ontologies. In addition, for reasoning, we used the FaCT++ that is the new generation of the well-known FaCT OWL-DL reasoner. FaCT++ uses the established FaCT algorithms, but with a different internal architecture.

As an experiment, we gave two different queries to find a service element as a scenario. 1) The agent semantically finds the service elements that are can provide cooling the room as illustrated in figure 7. 2) The agent considers an energy saving policy while providing same service with 1) as illustrated in figure 8. We defined number of services such as window, electric Fan, Air Conditioner and Electric Heater as the service model. In addition, we defined the policy of power consumption about appliances.

In case of 1), the reasoner searched possible elements


Query:

Query (class expression)

TempControl and hasCoolingPossibility some CoolingPossibility

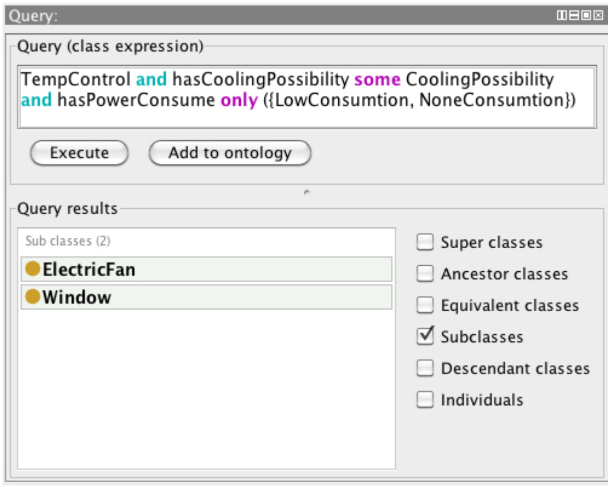
Query results

Sub classes (3)
- Aircon
- ElectricFan
- Window

Figure 7. A result of the temperature control serivce

such as Air Conditioner, Electric Fan and Window. Since the window represented its functionality on the semantic, the agent could infer that the window has functionality of the cooling. It's meaning that agent can understands diversified functionality of services according to the service object.

As a result of case 2), we can see that Air Conditioner is excepted in query results by considering of power consumption. Thus, a priority of service elements could be



TempControl and hasCoolingPossibility some CoolingPossibility and hasPowerConsume only ({LowConsumtion, NoneConsumtion})

- ElectricFan
- Window

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☑ Subclasses
☐ Descendant classes
☐ Individuals

Figure 8. A result of the temperature control serivce with the consumption policy

changed by considering the policy. We believe that our system provides to infer a relation between services and policy.

Since there are many semantic representations according to services, to represent all meaning in a single representation is actually impossible. The name space of ontology solves such problems by combining the different ontology model. Our system is able to include an external ontology model by importing its name space. Such distributed development of a service model will improve scalability of the service model. In the near future, a distributed environment for developing home service is going to be needed by service provider. Our system provides a platform to deploy a service that was made by an external service provider into the home network environment smoothly.

V. Conclusion

As we discussed, service composition needs to dynamically change according to service objective and various policies in the home network environment. In this paper, we proposed the dynamic service composition using Ratiocination on integrated home network system. Our system provides machine-readable service representation to understand the semantics of services and the relation between these services. It is also possible to provide a distributed developing environment using a name space.

We believe that our research provides the solution to decide a service semantically according to service objective considering the priority on the ground of a policy. Furthermore, since there are many semantic representations according to services and policies, to represent all meaning in a single representation is actually impossible. The system that used an ontology model can solves the challenge by importing the policy model from external service providers.

However, we still have several challenges such as suitable ontology mapping between policy models and service models, automated collecting of a policy which is needed when agent decide a service.

Acknowledgment

We would like to give one's thanks to Tan's laboratory members whom gave any little comment about this paper.

References

[1] Dutta-Roy, A., (December) Networks for Homes IEEE spectrum 36(12):26-33.

[2] Miller, B., Nixon, T., Tai, C., and Wood, MD., Home networking with universal plug and play. IEEE Commun Mag 39(12):104-109.

[3] Microsoft. UPnP Forum Web site, Microsoft Corporation, http://www.upnp.org/ [accessed: Mar 22, 2011]

[4] DLNA. digital living network alliances web site, DLNA overview and vision whitepaper 2007, http://www.dlna.org/ [accessed: Mar 22, 2011]

[5] Marples, D. and Kriens, P., (2001,December) The open services gateway initiative: an introductory overview. IEEE Commun Mag 39(12):110-114.

[6] ECHONET. web site ECHONET Specification Ver2.11 (English), http://www.echonet.gr.jp/ [accessed: Mar 22, 2011]

[7] Chakraborty, D. and Joshi, A. Dynamic Service Composition: State-of-the-Art and Research Directions,Technical Report TR-CS-01-19, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, USA, 2001.

[8] Mennie, D. and Pagurek, B., An Architecture to Support Dynamic Composition of Service Components, Proceedings of the 5th International Workshop on Component-Oriented Programming(WCOP 2000), Sophia Antipolis, France, 2000.

[9] Casati, F., Ilnicki, s., Jin, L., Krishnamoorthy, V and Shan, M., Adaptive and dynamic service composition in eFlow, In Proc. of the Int. Conference on Advanced Information Systems Engineering(CAiSE), Stockholm, Sweden, 2000.

[10] Chandrasekaran, S., Madden, S. and Ionescu, M., Ninja Paths: An Architecture for Composing Services over Wide Area Networks, CS262 class project write up, UC Berkeley, 2000.

[11] Fujii, K. and Suda, T., Dynamic Service Composition Using Semantic Information, ICSOC'04, November 15-19, New York, USA, 2004.

[12] OWL. web ontology language web site, OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/ [accessed: Mar 22, 2011]

[13] OWL-S. Semantic Markup for Web Services, a white paper describing the key elements of OWL-S, http://www.ai.sri.com/daml/services/owl-s/1.2/overview/ [accessed: Mar 22, 2011]

A Methodology for Household Appliances Behaviour Recognition in AmI Systems

Ovidiu Aritoni
E-Austria Research Institute
Timisoara, Romania
oaritoni@info.uvt.ro

Viorel Negru
West University of Timisoara
Department of Computer Science
Timisoara, Romania
vnegru@info.uvt.ro

***Abstract*—The actual research in green computing tries to understand the human behavior and how appliances are used in households. We propose, in this paper, a software prototype which can be used to understand the household appliances' behavior. Our architecture uses heuristic functions and pattern matching mechanisms in order to achieve this goal.**

***Keywords*-curent-cost sensor data-stream, appliance recognition,behavior recognition, heuristics.**

I. Introduction

Ambient intelligent systems (AmI) are characterized by the possibility to develop intelligent solutions for the contextual problems perceived from the ambient environment. Using data from various sensors, which are collected over a long time period, these systems can understand the ambient environment and they try to solve problems that may appear. The process of understanding the ambient environment does not mean the simple collection of data from the various sensors, this process sometimes necessitates the aggregation of data and the recognition of certain special behaviors. The identification of activities is done, in most cases, using data from several types of sensors [1]. For example, if the system try to recognize whether a person makes a sort of physical exercises, data from the following sensors types are used: EKG, temperature, proximity sensors, accelerometer, light sensors etc. To successfully recognize the human behavior, various techniques are used, this field of computing research being the main target of numerous studies. The Hidden Markov Model (HMM) is a probabilistic model used in the process of recognizing activities with the help of data collected from various sensors [2], [3], [4]. The activities may also be recognized using other techniques such as database exploration, data correlation and other data mining techniques [5]. This article will be focused on the understanding of ambient environment using the identification of certain behaviors. In the research project DEHEMS [6], the contextual understanding of the ambient environment comes from the data collected on the usage of household appliances. This usage is being monitored by the use of a single type of sensor namely a sensor designed to measure alternative curent (AC) current consumption. The system uses the data stream, from the AC sensor, to recognize and analyze the household appliances behavior.

The paper presents an ”intelligent” current-cost sensor which recognizes the scenario about how the appliances from a house are running. We concentrate on the next section to determine on positioning the paper in the research area of intelligent curent-cost sensors. Thereafter, we present the architecture and the main idea of our system. We conclude the paper with the important fields of the future work and a discussion about how to validate our system.

II. Related Work

The Bit-Watt system is used to recognize the appliances from a household, using in this process the devices power signature (previously known) [7]. The system presented in this article in contrast to the system presented in [7] will be able to identify the appliances connected to the household power grid. The Bit-Watt system's sensor is placed between the appliance and the electric socket, thus being able to identify only the appliance on which the sensor is placed.

Our study uses only one AC sensor that generates one stream of data which represents the power consumption of all the appliances from the entire house. There are no other sensors as in the Bit-Watt system, thus, we use only one sensor for the entire household. The purpose of this study is to determine the state in which all the household appliances are at any given time with the help of the data stream from the AC sensor.

The proposed system result is a scenario that describes the behavior of all the appliances from a household. It is important to know when and how often is used a specific appliance. The purpose of the DEHEMS project [6] is to develop recommendations about how to improve the home energy efficiency. An objective of the project is to detect the human actions or the users habits that increase the energy usage inefficiency. Using the appliances' behavior we can obtain information about the human behavior: the output of our system is used for the household residents behavior recognition. The human behavior analysis and the correlation with the appliances behavior help us to detect the actions or habits that reduce the energy efficiency. The statistics about the appliances most used in a household, are also useful in the recommendation process. The DEHEMS project [6] is part of the open-living labs experience [8]. We use the feedback of the household residents in order to improve our system, and the sensors data received from the living-labs houses.

III. Architecture

Our system has to know in advance the models of all the appliances from a household. This will enable to determine the various scenarios of current consumption. The definition of this scenario of current consumption is realized by identifying each appliance in turn from the single data stream being sent by the AC sensor. Another problem that needs to be addressed is that of subdividing the data stream from the AC sensor into specific data sets, for each appliance.

During the process, a repository of signatures for the different appliances present in the household will be used. The construction of this repository of power signatures can be made using empirical data by measuring the power signatures for each appliance in part. A signature is described using a finite-state machine. We build the repository using patterns recognition techniques on the data [9]. The discovery of a certain pattern repeating cyclically will lead to the conclusion that we have an appliance described by this pattern. These patterns will be stored in the system and will be used in the process of analyzing the behavior of household appliances.

Our system needs to know the producer and type of all the appliances from the household. Another existing approaches use a smart current-cost sensor to acquire data about the individual appliance. This means that we have to install numerous devices if we want to monitor the states of all the appliances. In this situations the cost of current-cost sensors network installation and maintenance will be higher than in our proposal. Our goal is to propose an ”intelligent current-cost sensor” which minimize the cost of deployment, using a single sensor instead of more sensors. Each household has a current-cost sensor that provides monthly the total power consumed. Our system uses this current-cost sensor or a special wrapper-sensor.

Also, our system uses a ”central” repository that stores the signatures of all the appliances. The proposed system is a module of a recommendation system for the energy efficiency improvement. This recommender system consists of two parts: a central-server and local subsystems for each household. The server-side provides web-services for data-storage about each household. A central database stores data about each household: the number of appliances, a list of all the appliances, appliances signatures, the current-cost sensor datastream, and other important information. The amount effort that is required to train our system is reduced using the central-server. Also, the central server is useful when we want to compare the appliances behavior and help us to detect the devices’ malfunctions. For each household we have an instance of the proposed system.

In order to describe successfully the power consumption scenarios it is necessary to decompose the primary data stream into smaller data sets which describe the state in which a particular appliance is. To accomplish this, we use a Greedy algorithm [10] and matching mechanisms. Every value of the data stream will be decomposed into other values, this process being an ongoing one within the system. The values resulting through decomposition and the different power signatures the matching coefficient is computed which in turn will be used to detect which appliance is in use and what is current its state. Greedy decomposition of certain values will be made in the inflexion points of the data stream. The points situated before and after the inflexion point will be used to check the conclusion reached from the analysis of the inflexion point. One or more tuples will result from the data stream decomposition. Constructing and generating associations between these tuples will be realized using heuristics which are determined empirically from real world. For example some appliances can have a predictable and linear behavior, one example are refrigerators which switch on and off a few minutes at a time depending on the setting and outside temperature.

The present application achieves a breakdown of the total electric consumption of devices that can operate in a house. We use a heuristic function defined by people at home through a questionnaire. The heuristics are used only when there is uncertainty in the detection pattern (eg. several matching patterns). Our model is based on heuristic functions that help us to reduce the problem complexity. A heuristic function returns for a specified time moment a list of devices that can run at the moment and a probability for each device to run at this time:
$h : T \longmapsto \chi, h(t) = \{(d_1, p_1), (d_2, p_2), ..., (d_n, p_n)\}$
where:
t is a time interval
d_i is a device
p_i is the probability to use the device d_i at that moment.
$\chi = \Delta \times P$, Δ is the ensemble of all the devices and p_i is a probability, $p_i \in P = [0, 1]$
Also we use another heuristic functions, likely the association rules:
$h^* : T \times \Delta \longmapsto \chi$
$h(t, D) = \{(d_1, p_1), (d_2, p_2), ..., (d_n, p_n)\}$
The h^* heuristic inform us that if at the current moment the device D is running, after, on the t time interval can run the devices $d_i, i = 1, ..., n$.

The heuristic values are based on a questionnaire. The users will answer to a lot of questions about their living style. Using these answers we calculate the heuristic values. The heuristic function and the use of the central ”repository” of appliances signature provide a large scalability for our system.

The purpose of this study is to construct scenarios regarding the usage of household appliances using the data stream from the AC sensor mounted on the household electrical grid. These scenarios will specify: the order in which

appliances are running, the moment when an appliance starts, the time moment when an appliance stops running and the moment when an appliances normal functioning is interrupted by certain events. An example of such scenario is:
7.00 LightBedroom ON
7.02 LightBathroom ON
7.08 LightBeedrom OFF 100%
7.08 CoffeeMaker ON
7.12 LightBathroom OFF 100%
7.12 Refrigerator ON
7.16 CoffeMaker OFF 87%
....
10.22 PEAK
....
As one can see, the light from bedroom is ON at 7.00 and OFF at 7.08, and we conclude that the light was run normally 100%. The coffee maker runs from 7.08 to 7.16, but it has run only 87% normally. This coefficient is calculated as the similarity factor between the nominal time-serie (from the repository) and the actual time-serie. At 10.22 we have a PEAK event on the household electrical grid.

This information is necessary in a smart home, because it enables the system to respond to the inhabitants needs. Using these scenarios some predictions can be made by the AmI system. Using the predictions some improvements may be made that will increase the level of comfort and/or power usage efficiency of a smart home. For example, if the time when the user usually goes to sleep can be inferred, the AmI system can turn off all necessary household appliances during the time the user sleeps.

Our system uses as input the power readings stream of a user's device and produced as output: determination of the malfunctioning of the device (health status) and other statistics. Our first goal was to define a set of rules, so that, given a set of energy readings, would evaluate the health of a device and pinpoint regions of faulty power usage. A set of reading has at its core two most evident states: ON and OFF. This information developed our first rule in evaluating device consumption. Thus, before going any further, we must extract the actual usage that characterizes the ON state of the device. Further information has to be provided about each specific type of device in order to effectively determine health status, and such information has to be accessible. Given that, as a general rule, the correct function of a device, as observed only by consulting the values representing the power consumption, has to fit between some boundaries: the maximum power consumption and minimum power consumption, the former, as observed earlier, having to reflect the ON state of the device. The device specifications sometimes include the maximum and minimum power consumptions. However, for the purpose of this project, these were taken from energy usage patterns. Using our system we analyzed the power consumptions of our laptops and successfully created charts that show that the deviations are negative, which means the devices operate under the minimum power consumption specification, making it optimal.

IV. Conclusion and Future Works

We use in our study a current-cost sensor to collect data from usually devices such as refrigerators, computers, multimedia devices, etc. The heuristics are extracted via a questionnaire applied to more than 1000 inhabitants from rural and urban areas from the Banat region of Romania. The evaluation of the proposed prototype will be done using intelligent current-cost sensors for each appliance. These sensors will collect data for each appliance and in this way we will obtain a scenario for each appliance from the household. A sum of all these appliances scenarios will be compared with our results (the behavior of our appliances). We use a comparation between these scenarios for the evaluation of our prototype. Furthermore, we can use in the evaluation process a curent-cost sensor data simulator, in order to compare the simulation scenario with the output of our prototype.

Our software prototype uses finite-state machine as model for each appliance's behavior, and heuristic functions to reduce the complexity of the problem. This approach consists in observing the specific pattern of each device. After this, we identify these patterns in the total consumption using our heuristics.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement N^0 No224609 *DEHEMS*, and by the Romanian *DEHEMS-RO* project *RO-PNII-20-EU/11.06.09*. The views expressed in this paper do not necessarily reflect those of the DEHEMS consortium members.

References

[1] A. Kofod-Petersen and J. Cassens, "Explanations and context in ambient intelligent systems," in *CONTEXT'07: Proceedings of the 6th international and interdisciplinary conference on Modeling and using context*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 303–316.

[2] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*. New York, NY, USA: ACM, 2008, pp. 1–9.

[3] W. Huang, J. Zhang, and Z. Liu, "Activity recognition based on hidden markov models," in *KSEM'07: Proceedings of the 2nd international conference on Knowledge science, engineering and management*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 532–537.

[4] D. Sanchez, M. Tentori, and J. Favela, “Hidden markov models for activity recognition in ambient intelligence environments,” in *ENC ’07: Proceedings of the Eighth Mexican International Conference on Current Trends in Computer Science*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 33–40.

[5] S. Lühr, G. West, and S. Venkatesh, “Recognition of emergent human behaviour in a smart home: A data mining approach,” *Pervasive Mob. Comput.*, vol. 3, no. 2, pp. 95–116, 2007.

[6] “http://www.dehems.eu [retrieved: May 10, 2011].”

[7] T. Kato, H. S. Cho, D. Lee, T. Toyomura, and T. Yamazaki, “Appliance recognition from electric current signals for information-energy integrated network in home environments,” in *ICOST ’09: Proceedings of the 7th International Conference on Smart Homes and Health Telematics*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 150–157.

[8] “http://www.openlivinglabs.eu [retrieved: May 10, 2011].”

[9] S. E. Rombo and G. Terracina, “Discovering representative models in large time series databases,” in *FQAS*, 2004, pp. 84–97.

[10] A. R. Teixeira, A. M. Tomé, and E. W. Lang, “Greedy kpca in biomedical signal processing,” in *Proceedings of the 17th international conference on Artificial neural networks*, ser. ICANN’07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 486–495. [Online]. Available: http://portal.acm.org/citation.cfm?id=1778066.1778123

Collaboration Language for Social Information Engineering

Kurt Englmeier
Faculty of Computer Science,
P.O. Box 10 04 52, 98574
Schmalkalden, Germany
k.englmeier@fh-sm.de

Javier Pereira
Engineering faculty, Universidad Diego Portales,
Av. Ejército 441, Santiago, Chile
javier.pereira@udp.cl

Josiane Mothe
Institut de Recherche en Informatique de Toulouse,
UMR5505, CNRS, Université de Toulouse, IUFM
Toulouse, France
josiane.mothe@irit.fr

Duska Rosenberg
iCOM Centre for Research in Information, Computing
and Communication, Royal Holloway Univ. of London
London, UK
duskarosenberg@me.com

Fionn Murtagh
Dept. Computer Science, Royal Holloway,
University of London, Egham, Surrey TW20 0EX, UK
fmurtagh@acm.org

Abstract— **In this paper, we present a technology platform for personalised learning environments and discuss the key features that will enhance user experience. In particular, we consider elements of social software, integrated tools and user modelled services as they converge on the development of folksonomies. We base our discussion on design principles for accessing and sharing a range of different resources, tools and services for both individual and group learning. In the paper, we concentrate on the language model that supports the development of process-related folksonomies. The model emerges from both the functionality of the learning environment and the language applied in the specific learning environment, thus providing a common platform that users can themselves adapt to their specific learning needs and circumstances.**

Keywords- model of query language; social participation.

I. INTRODUCTION

Folksonomies or collaborative tagging became popular on the Web through social bookmarking and photograph tagging. This facet of social computing demonstrates that social activities on the web can form information landscapes. We think that folksonomies add a new quality to social computing: social information engineering. Our project Co-lingua reflects work-in-progress that fosters social tagging not only for particular information items but also for the information integration processes applied to generate information.

Information integration applications are prime candidates among systems that encourage active user engineering. Their requirements, wants, needs, ideas, and skills are so manifold and diverse that no traditional software engineering approach can capture them all in a satisfactory way. We therefore have to harness the power of user communities in order to share information.

With Co-lingua, we take up the rationale of technology-mediated social participation (TMSP) [3] and develop an open interaction language. "Open" means that users can adopt it as their commodity toolset, and also that its facets accommodate their domain-specific jargon. A recent study, conducted by Forrester Consulting [8], found that knowledge workers will significantly *benefit from a blended solution* that would allow them to fulfil most of their information requirements, while requiring minimal support from Information Technology (IT). While Forrester does not explain how such a blended solution may work, we are at least told that some minimal support from IT is one key ingredient of TMSP in the workplace. The users take the role of "prosumers", i.e., users frequently changing between the role of information consumers and information producers. Prosumers produce goods they and/or others consume.

Prosumer computing represents a blended engineering concept that emphasises collaboration between IT and users, where IT provides minimal support to users that develop self-service solutions autonomously. Co-lingua translates this rationale of prosumer computing into a new paradigm for information access and mining software: under-designed software fine-tuned by information access and mining prosumers using a controlled language interaction layer.

Within this rationale, the objective of Co-lingua is to transform natural language statements into machine-processable instructions [8]. These instructions control the *underdesigned software that provides generic features for search, access to structured and unstructured data, data cleansing, and data merging*. On top of that resides the controlled interaction language layer that lets users fine-tune the generic features using human language in all sorts of learning environments that direct user to develop suitable answers for questions like [11]:

- "What are the actual sales figures for the automobile industry in Greece?"
- "How can I build an indicator for the economic performance of the European industry?"
- "What kind of cancellation policy applies for booking BY2TF6?"

- "How can I join credit card information and customer demographics?"

In the second section, we present the rationale of our language model. Section 3 explains the semantics of the interaction model. Section 4 outlines how named entities and expressions of communicative acts are integrated through a domain-specific ontology. Section 5 concludes the paper.

II. RATIONALE AND RELATED WORK

A. *Scenario*

We take, for instance, an economic indicator system that helps people to learn more about indicators and to set up their own indicator system. Let us assume the user wants to compare the actual economic situation of the industry in the Maghreb states with that in the states of Southern Europe. The indicator is the product of a number of processing steps. From each country of the regions addressed three time series need to be retrieved: "business climate", "incoming orders", and "six months business expectations". According to the economists' advice, the user shall create composite time series for both regions. In the composite, each country has a different weight, of course. After developing moving averages (over 3 periods) from each time series, the user can compare the two data sets.

B. *Rationale*

Our concept of folksonomies covers three practical aspects that merge into a unified, domain-specific interaction language:

- The **service aspect** comprises first generic data access and processing services, the target of the learning environment. The huge diversity of economic indicators makes is impossible to fix all conceivable user requirements for the access and processing services. These services have to be generic, by nature, in order to accommodate the broad variety of their later use. The first part of the folksonomy results from "translation" of service signatures into expressions of the user community. This part can be a simplified command language similar to macro-languages. Moving averages over three periods of time series may result from an instruction like "moving_averages.time_series.3".
- Generic services need to be fine-tuned before they can be used. This can be achieved by **instructional folksonomies**. These come closer to the language action perspective explained below. They instruct, for instance, that "in any aggregate of Maghreb industrial data, Moroccan data enter with a weight of 1.8". The language applied to cover instruction aspects can also be a simplified command language like "economic situation = business climate + incoming orders".
- The **action language aspect** addresses primarily the users. Their actions reflect their navigation in the learning environment and the use of the tools provided. Their statements, thus, reflect directives, rejections, requests, and the like: "for aggregation apply 6-period moving averages", "send me the latest figures on business climate in Turkey when available", etc.

C. *Related work*

User statements, emerging from the instructional or language action aspect, usually contain precise and clear descriptions of resources and activities required to complete tasks. Merging language elements from the three aspects mentioned above lead to a controlled vocabulary [7] that semantically covers all user interaction possible in this particular environment [2].

Communicative acts performed during interaction reflect the users' domain expertise and their intentions. Our language model follows the rationale of *Language Action Perspective (LAP)* as developed in [4], for instance. LAP has roots in *speech act theory* [12] and the *conversation-for-action (cfa) schema* introduced by Winograd and Flores [14] (see also [1]). There were a number of approaches emerging from this schema, the *Action Workflow* [14], for instance, or the *BAT (Business interAction and Transaction) model* [4].

The rationale "is to get a model of how people, through conversation, coordinate their work" [13]. The language used in the communicative acts represents the language for the definition and coordination of processes. A storybook serves as a means for editing, categorising, and linking of the language elements. Communicative acts are developed around the storybook metaphor and the principles of narration. It forms the blueprint for Co-lingua's interaction model.

III. SEMANTICS OF THE INTERACTION MODEL

Folksonomies in Co-lingua are constrained by semantic representations of the functionality of the respective learning environment. The controlled language thus covers all use patterns related to this functionality. Conversely, all variants of interactions and objects that can be associated with the functionality form the semantic space the controlled vocabulary has to cover. In principle, an ontology for the specific learning environment could constitute the centrepiece of this interaction language. The ontology needs to be grounded sufficiently in the signature of the services. It includes the roles of the actors involved, the objects addressed, and the objects' collaboration model. In combination with the ontology, the interaction model defines and controls not only user dialogues but also service choreography and thus aligns work flow and flow of information.

The challenge is the open definition of the interaction language that maps to information integration functionality. Definitions must be "open" in terms of language elements being individually definable and adaptable by the users. The learning environment, in turn, defines the functionality. The target language is the language users expect when interacting in natural language with a learning environment. The development of the target language follows first a bottom-up approach. The starting point is the source language emerging from generic services. Through gradual enhancement, the source language develops into the simplified command language.

The *service interfaces* provide an initial set of elements for the source language. Each service hosts a number of actions referring to a particular theme of task. A *search* action, for instance, looks for one or more *items* in one or more *containers*, while each item to be included may have one or more characteristic *attributes*. This stereotypical search pattern appears in SQL statements as well as text search. The source language is thus a unified command language the services can interpret. In a very simplified version such a command language can take the form

search.archive_of_persons.person.age > 24,
country.residence=greece,

for looking for persons older than 24 years and living in Greece. By adapting the nomenclature that gains popularity in programming languages like Ruby we say that the plural of a noun automatically indicates the archive containing items named by that noun. For example, the archive "persons" contains items of type "person". This reduces the command to search.persons.age > 24, country.residence=Greece.

We emphasise that this language is certainly an oversimplified version for a stereotypical action pattern that is probably better represented in WSDL (Web Service Definition Language). However, it suffices to explain the first level of the source language.

At a first glance, this command language is already a standard language IT people should agree upon. However, in many cases such a standard exists in IT. In particular when IT departments apply semantic web standards, they enforce standardised interface and object representations. The collaboration already in place in many IT departments and the enforcement of common standards supports the take-up of the collaborative management of a standard command language.

Our *simplified command language* is first a proposal for an intuitive language on the interface level of services. It is intuitive, because it follows commonly known syntax patterns of programming languages. Although quite simple, this command language is powerful and easy to implement. A parser uses a series of regular expressions to interpret the command. The assignments of command terms to action patterns can be explicit or implicit through nomenclatures. Users define assignments by explicit statements like container=archive_of_persons, persons, employees or country. residence="RESIDENCE" (to adapt different column names to the command languages).

Co-lingua's determination of language concepts uses grammar-free text analysis. It applies methods for automatic text analysis such as stopword elimination, identification of composed terms, and named entity recognition. Recursive pattern analysis develops a hierarchical structure of language concepts:

- It starts with determining basic elements like word, numerical value, date, address, value range, booking code, zip-code, email address, etc. through Regular Expression analysis.
- Next, we repeat pattern analysis on the term level to determine compounds of basic elements reflecting minimal semantics like composed terms including combinations with numerical values ("older than 24", "weight: .35", etc.). Composed term determination precedes this analysis step.
- Minimal semantic expressions are input to further pattern analysis. Like in determining composed terms we repeat the determination of terms frequently appearing in close proximity. We expect to reveal superior concepts like "birth" identified in patterns like "[name], born in [name] on [date]". This determination process will see human intervention to assign concept names to term patterns.
- The next step is the categorisation of the concepts identified in the previous step. Categorisation here means both structuring concepts along generalisation-specialisation as well as hierarchical annotation of text documents according to the hierarchical organisation of the text (along titles, subtitles, paragraphs, etc.).

The end result is a thesaurus developed along categorised named entities with ramifications into text documents and into command language expressions. It is important to note, that the first candidates for named entities are the parameters of the service together with their names (i.e., the terms applied to express their signatures.). This thesaurus represents the named entities of the interaction language.

IV. METALANGUAGE LAYER FOR SOCIAL INFORMATION ENGINEERING

It is certainly reasonable to model social information engineering along an ontology that processes (task ontology) and actors and objects (onto terminology). The conceptual model is the blueprint for domain-specific meta-models. The Co-lingua metalanguage should preferably be relatively simple and precisely specified so as to be amenable to processing by the generic information integration services. In principle, this is the approach taken in the semantic web, where ontologies are used to provide extensible vocabularies of terms.

The interaction language shall be powerful enough to enable users, by following their intuition, to define classes, properties and relationships while the system checks for implicit subsumption relationships and gives feedback about the logical implications of their design, including warnings about inconsistencies and synonyms. Ontologies allow the representation of semantic memory, but a narrative is a way in which a story is told. Finally, the dialogue system managing the storybooks will be able to explain inferences; without explanations, users may find it difficult to repair errors in the ontology and may even start to doubt the correctness of inferences. Argument schemas may contribute to provide explanations by analysis of arguments for and against some claim (e.g., a proposition, an action intention, a preference, etc.). An argument schema consists of a claim, a set of premises, i.e., a number of supportive and possible denying reasons, and an aggregation mechanism to reach a global conclusion concerning the validity of that claim [10].

By applying the social computing paradigm it is possible for user communities to collaboratively create simple forms of ontology via the development of action signatures and object/actor tags organised in folksonomies [13]. The ontology plays the following roles: provides a formally defined extensible vocabulary for semantic annotations; describes the structure of domain sources and the information they store; and provide detailed model of the domain against which queries are formulated. The semantics of the vocabulary is mainly captured informally in textual and contextual descriptions of each term and procedurally interpreted by information integration services. This informality reduces the complexity of reasoning without limiting the ability of the services to “understand” vocabularies. It enables, in any case, logic for which query answering can be implemented using rule-based techniques.

OWL, the Web Ontology Language, has been defined to be used for web resource indexing. Thanks to OWL, it is possible to represent the lexicon in the form of which a concept will occur in a document or the user’s interaction. To model the terms used for this concept, OWL associates with the associated class one or several sequences of characters by the means of RDF-S.

IRIT laboratory developed a model that aims at considering context, content and communicative acts in information search and navigation related to learning. We propose to model two main aspects of context: the themes of the user’s problem and the specific information the user is looking for, to achieve a particular learning task. Both aspects are modelled by means of ontologies. Documents are semantically indexed according to the context representation and the user accesses information by browsing the ontologies. OntoExplo is an interface we developed based on this model. It has been applied to a case study in the domain of astronomy that has shown the added value of such a semantic representation of context [5] as well as in the domains of e-learning [6] and security.

Figure 1 presents the OntoExplo interface. The documents that are handled are scientific papers in the domain of public health. The task ontology is presented on the left hand side of the figure. It corresponds to meta-data associated with the documents (authors, title, language, etc.). This description follows the Dublin Core. On the right hand side of the figure, the security domain-ontology is presented. It can be browsed to see the various concepts. In addition, the user can query the ontology to access a specific concept.

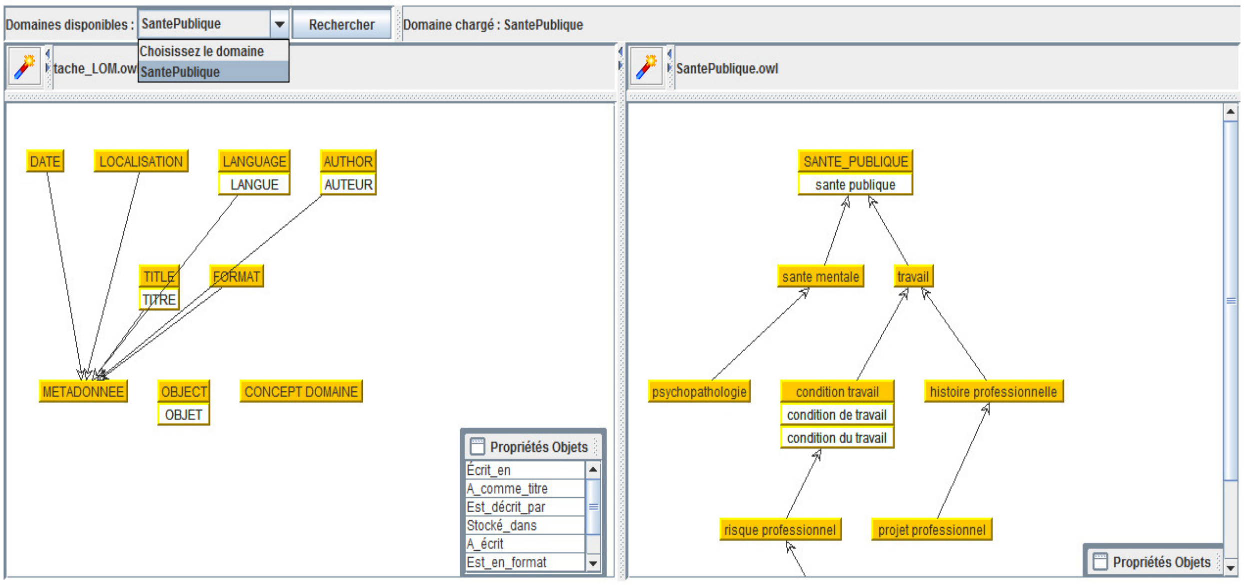
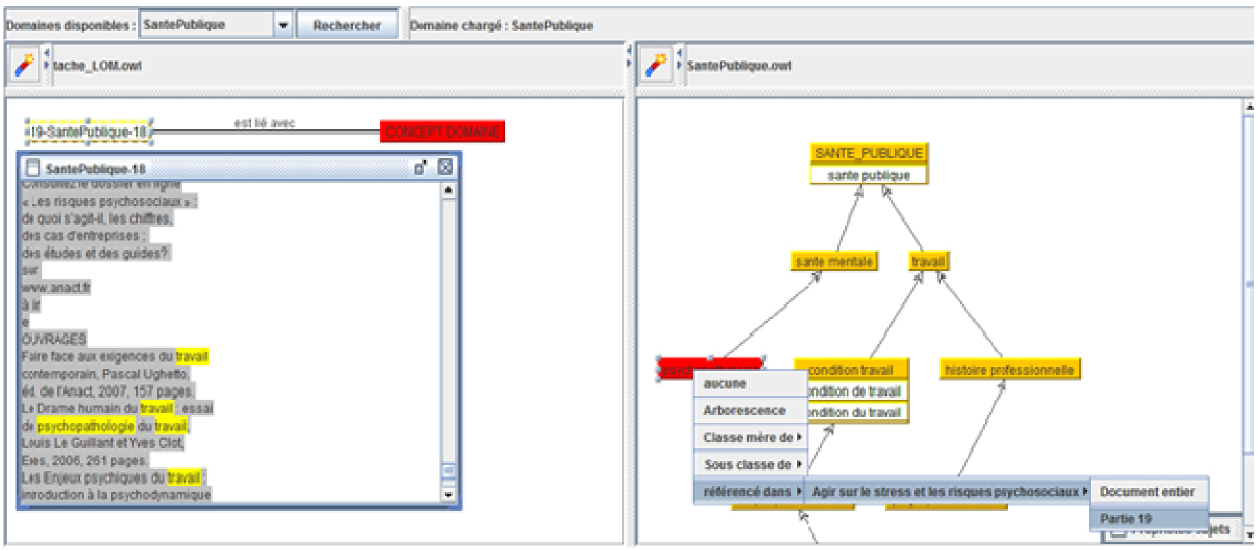


Figure 1. OntoExplo interface, task and domain ontologies.

One of the documents is more specifically visualised on the left hand side of Figure 2. Its description is visualised. From this, it is then possible to interactively visualise the concepts from the domain ontology that occur in this specific document. Task and domain ontologies can be browsed the other way around: the user can visualise the documents that are indexed by a specific domain concept that is chosen. Alternatively, she can select an author name and dynamically visualise the set of documents this author wrote.



Consultez le dossier en ligne
« Les risques psychosociaux » :
de quoi s'agit-il, les chiffres,
des cas d'entreprises ;
des études et des guides?
sur
www.anact.fr
à lir
e
OUVRAGES
Faire face aux exigences du travail
contemporain, Pascal Ughetto,
éd. de l'Anact, 2007, 157 pages.
Le Drame humain du travail : essai
de psychopathologie du travail,
Louis Le Guillant et Yves Clot,
Eres, 2006, 261 pages.
Les Enjeux psychiques du travail :
introduction à la psychodynamique

Figure 2. OntoExplo interface, browsing task and domain ontologies.

V. CONCLUSION AND FUTURE WORK

As stated above, this paper presents work in progress, representing the development of technology-mediated social participation approach suitable for user-centred design of personalised learning environments. Both education and industry have recognised the benefits of platforms supporting social activity in order to enrich our everyday work and to make our work processes more efficient. This demonstrates the need for stronger focus on social computing in the design of collaborative environments for learning and work. However, the successful development of new applications along the social computing paradigm also demonstrates that the community of "engineers" is not limited by IT skills alone. Users are excellent "engineers" when it comes to organising their information processes. The first step is to endow them with an information engineering environment that enables them to express through folksonomies how information integration should behave in their individual domains. Our future work will illustrate a way in which folksonomies guide the entire process of developing a technology platform that supports both individual and social learning.

VI. REFERENCES

[1] Brown, P.C., Succeeding with SOA: Realizing Business Value Through Total Architecture, Addison Wesley Professional, Boston, MA, 2007.

[2] Dietz, J.L.G., The Deep Structure of Business Processes, Communications of the ACM, vol. 49, n. 5, 2006, pp. 59-64.

[3] Fruchter, R, Nishida, T., and Rosenberg, D., Understanding mediated communication: the social Intelligence Design approach, AI & Society-Social Intelligence Design for Mediated Communication, vol. 19, n. 1, 2005, pp. 1-7.

[4] Goldkuhl, G., Action and Media in Interorganizational Interaction. Communications of the ACM, vol. 49, n. 5, 2006, pp. 53-57

[5] Hernandez, N., Mothe, J., Chrisment, C., and Egret, D., Modeling context through domain ontologies, Journal of Information Retrieval, vol. 10, n. 2, 2007, pp. 143-172.

[6] Hernandez, N., Mothe, J., Ralalason, B., Ramamonjisoa, A., and Stolf, P., A Model to Represent the Facets of Learning Objects, Interdisciplinary Journal of E-Learning and Learning Objects, vol. 4, 2008, pp. 65-82.

[7] Huijsen, W.-O., Controlled Language—An Introduction, Proc. 2nd Int'l Workshop Controlled Language Applications, Carnegie Mellon Univ., 1998, pp. 1-15

[8] Englmeier, K., Pereira, J. Mothe, J., Choregraphy of web services based on natural language storybooks, international conference on Electronic commerce, 2006, pp. 132-138.

[9] Lean Business Intelligence, Forrester Consulting, October 2, 2009. http://www.slideshare.net/findwhitepapers/lean-business-intelligence-how-and-why-organizations-are-moving-to-selfservice-bi, 31 march 2011.

[10] Ouerdane, W., Maudet, N., and Tsoukias, T., Argument Schemes and Critical Questions for Decision Aiding Process. In Proceedings of the 2nd International Conference on Computational Models of Argument, 2008, pp. 285-296, IOS Press.Toulouse - France,.

[11] Ravat, F., Teste, O., Tournier, R., and Zurfluh, G., Graphical, Querying Multidimensional Databases. 11th East-European Conference on Advances in Databases and Information Systems, Springer-Verlag, LNCS 4690, 2007, pp. 298-313, Varna (Bulgarie).

[12] Searle, J.R., Speech Acts – An Essay in the Philosophy of Language, Cambridge University Press, Cambridge, MA, 1969.

[13] Spyns, P., de Moor, A., Vandenbussche, J., and Meersman, R., From folksologies to ontologies: How the twain meet. In Proceedings of On the Move to Meaningful Internet Systems 2006: CoopIS, DOA, GADA, and ODBASE, Lecture Notes in Computer Science 4275, Springer, 2006, pp. 738–755.

[14] Winograd, T. and Flores, F., Understanding Computers and Cognition: A New Foundation for Design. Ablex, Norwood, 1986.