Studies in Computational Intelligence 315

Mohammad Essaaidi
Michele Malgeri
Costin Badica (Eds.)

Intelligent Distributed Computing IV

Proceedings of the 4th International Symposium on Intelligent Distributed Computing - IDC 2010, Tangier, Morocco, September 2010

Springer

Mohammad Essaaidi, Michele Malgeri, and Costin Badica (Eds.)

Intelligent Distributed Computing IV

Studies in Computational Intelligence, Volume 315

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Mohammad Essaaidi, Michele Malgeri, and
Costin Badica (Eds.)

Intelligent Distributed Computing IV

Proceedings of the 4th International Symposium on Intelligent Distributed Computing - IDC 2010, Tangier, Morocco, September 2010

Springer

Prof. Mohammad Essaaidi
Electronics and Microwaves Group
Faculty of Science
Abdelmalek Essaadi University
P.O. Box 2121
Tetuan 93000
Morocco
E-mail: essaaidi@ieee.org

Prof. Michele Malgeri
Dipartimento di Ingegneria Informatica
e delle Telecomunicazioni
Università di Catania
Viale Andrea Doria 6
Catania
Italy
E-mail: mmalgeri@diit.unict.it

Prof. Costin Badica
Software Engineering Department
Faculty of Automatics.
Computers and Electronics
University of Craiova
Bvd. Decebal Nr. 107
200440 Craiova
Romania
E-mail: badica_costin@software.ucv.ro

ISBN 978-3-642-15210-8 e-ISBN 978-3-642-15211-5

DOI 10.1007/978-3-642-15211-5

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2010932507

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

The emergent field of Intelligent Distributed Computing brings together two well established areas of Computer Science: Distributed Computing and Computational Intelligence. Theoretical foundations and practical applications of Intelligent Distributed Computing set the premises for the new generation of intelligent distributed information systems.

Intelligent Distributed Computing – IDC Symposium Series was started as an initiative of research groups from: (i) *Systems Research Institute, Polish Academy of Sciences in Warsaw, Poland* and (ii) *Software Engineering Department of the University of Craiova, Craiova, Romania*. IDC aims at bringing together researchers and practitioners involved in all aspects of Intelligent Distributed Computing. IDC is interested in works that are relevant for both Distributed Computing and Computational Intelligence, with scientific contribution in at least one of these two areas. IDC'2010 was the fourth event in the series and was hosted by *IEEE Morocco Section* in *Tangier, Morocco* during September 16-18, 2010. IDC'2010 was collocated with: (i) NATO summer school on "Software Agents, Agent Systems and their Applications"; (ii) 2^{nd} International Workshop on Multi-Agent Systems Technology and Semantics, MASTS'2010.

The material published in this book is divided into three main parts: (i) 1 invited contribution; (ii) 28 contributions of IDC'2010 participants; and (iii) 4 contributions of MASTS'2010 participants.

The response to IDC'2010 call for paper was generous. We received 53 submissions from 20 countries (we counted the country of each coauthor for each submitted paper). Each submission was carefully reviewed by at least 3 members of the IDC'2010 Program Committee. Acceptance and publication were judged based on the relevance to the symposium themes, clarity of presentation, originality and accuracy of results and proposed solutions. Finally 17 regular papers and 11 short papers were selected for presentation and were included in this volume, resulting in acceptance rates of 32.07 % for regular papers and 52.83 % for regular and short papers. Additionally, MASTS'2010 workshop received 8 submissions. After the careful review (each submission was reviewed by at least 3 members of the MASTS'2010 Program Committee), 4 regular papers were selected for presentation and were included in this book.

The 33 contributions published in this book address many topics related to theory and applications of intelligent distributed computing and multi-agent systems, including: agent-based e-business, ambient intelligence, bioinformatics, collaborative systems, cryptography and security, distributed algorithms, distributed data mining, distributed simulation, grid computing, image processing, information extraction, knowledge management, logic programming, mobile agents, ontologies, pervasive computing, Petri nets, process modeling, self-organizing systems, semantic web services, service oriented computing, social networks and trust, swarm intelligence, web applications, wireless sensor networks.

We would like to thank to Prof. Janusz Kacprzyk, editor of *Studies in Computational Intelligence* series and member of the Steering Committee for their kind support and encouragement in starting and continuing the IDC Symposium Series. We would like to thank to the IDC'2010 and MASTS'2010 Program Committee members for their work in promoting the event and refereeing submissions and also to all colleagues who submitted papers to IDC'2010 and MASTS'2010. We deeply appreciate the efforts of our invited speaker Prof. Amal EL Fallah Seghrouchni and thank her for the interesting lecture. Special thanks also go to organizers of MASTS'2010: Adina Magda Florea, Amal El Fallah Seghrouchni, and Viorel Negru. Finally, we appreciate the efforts of local organizers on behalf of IEEE Morocco Section for organizing and hosting IDC'2010 and MASTS'2010.

Tangier, Catania, Craiova
June 2010

Mohammad Essaaidi
Michele Malgeri
Costin Bădică

Organization

Organizers

IEEE Morocco Section
Software Engineering Department, Faculty of Automation, Computers and Electronics, University of Craiova, Romania

Conference Chair

Mohammad Essaaidi	Abdelmalek Essaadi University, Morocco

Steering Committee

Janusz Kacprzyk	Polish Academy of Sciences, Poland
Costin Bădică	University of Craiova, Romania
Michele Malgeri	Università di Catania, Italia
George A. Papadopoulos	University of Cyprus, Cyprus
Marcin Paprzycki	Polish Academy of Sciences, Poland

Invited Speaker

Amal EL Fallah Seghrouchni	Université Pierre et Marie Curie & CNRS, France

Program Committee Chairs

Michele Malgeri	Università di Catania, Italia
Costin Bădică	University of Craiova, Romania

Program Committee

Răzvan Andonie	Central Washigton University, USA and University of Transylvania, Romania
Amar Balla	LCMS-ESI Ecole nationale Supérieure d'Informatique, Algeria
Nick Bassiliades	Aristotle University of Thessaloniki, Greece
Doina Bein	ARL, Pennsylvania State University, USA
Svetla Boytcheva	State University of Library Studies and Information Technologies (SULSIT), Bulgaria
Frances Brazier	Delft University of Technology, Netherlands
Dumitru Dan Burdescu	University of Craiova, Romania
Giacomo Cabri	University of Modena and Reggio Emilia, Italy
David Camacho	Universidad Autonoma de Madrid, Spain
Vincenza Carchiolo	University of Catania, Italy
Jen-Yao Chung	IBM T.J. Watson Research Center, USA
Gabriel Ciobanu	"A.I.Cuza" University of Iaşi, Romania
Phan Cong-Vinh	Centre for Applied Formal Methods, London South Bank University, UK
Valentin Cristea	"Politehnica" University of Bucharest, Romania
Paul Davidsson	Blekinge Institute of Technology, Sweden
Beniamino Di Martino	Second University of Naples, Italy
Luminiţa Dumitriu	"Dunarea de Jos" University of Galaţi, Romania
George Eleftherakis	CITY International Faculty, University of Sheffield, Greece
Vadim A. Ermolayev	Zaporozhye National University, Ukraine
Adina Magda Florea	"Politehnica" University of Bucharest, Romania
Maria Ganzha	Elblag University of Humanities and Economics, Poland
Adrian Giurca	Brandenburg University of Technology at Cottbus, Germany
Nathan Griffiths	University of Warwick, UK
De-Shuang Huang	Chinese Academy of Sciences, China
Mirjana Ivanović	University of Novi Sad, Serbia
Jason J. Jung	Yeungnam University, South Korea
Barbara Keplicz	Institute of Informatics, Warsaw University and Institute of Computer Science, Polish Academy of Sciences, Poland
Igor Kotenko	Russian Academy of Sciences, Russia
Dariusz Krol	Wroclaw University of Technology, Poland

Additional Reviewers

Marius Brezovan	Yann Krupa
Pasquale Cantiello	Niklas Lavesson
Alexandru Cicortaş	Angela Locoro
Nicolae Constantinescu	Georgios Meditskos
Mirel Coşulschi	Francesco Moscato
Marian Viorel Craciun	Milos Radovanovic
Marcin Dziubiński	Catalin Stoean
Heinz Erbe	Alina Strachocka
Sebastian Ernst	Michal Ślizak
Mihai Gabroveanu	Martijn Warnier
Peter Kopacek	Reda Yaich
Kalliopi Kravari	

Organizing Committee

Mohammad Essaaidi	Abdelmalek Essaadi University, Morocco
Badr Eddine El Mohajir	Abdelmalek Essaadi University, Morocco
Dumitru Dan Burdescu	University of Craiova, Romania
Amelia Bădică	University of Craiova, Romania

Advertising Chairs

Elvira Popescu	University of Craiova, Romania
Mihnea Scafeş	University of Craiova, Romania

MASTS'2010 Workshop Chairs

Adina Magda Florea	"Politehnica" University of Bucharest, Romania
Amal EL Fallah Seghrouchni	Université Pierre et Marie Curie & CNRS, France
Viorel Negru	Western University of Timişoara, Romania

MASTS'2010 Workshop Program Committee

Costin Bădică	University of Craiova, Romania
Valentina Balas	"Aurel Vlaicu" University of Arad, Romania
Matteo Baldoni	University of Torino, Italy
Olivier Boissier	ENS des Mines Saint-Etienne, France
Vincent Corruble	Université Pierre & Marie Curie, France

Contents

Part I
Invited Papers

Multi-Agent Systems: A Paradigm to Design Ambient Intelligent Applications

Amal El Fallah Seghrouchni, Adina Magda Florea, and Andrei Olaru

Abstract. In this paper we present a Multi-Agent System (MAS) paradigm and discuss how it can be used to design intelligent and distributed systems. The main features of this MAS, such as natural distribution of the system, inherent intelligence of its agents, and their mobility help address a large scope of distributed applications including the domain of ambient intelligence. Other features of the MAS, like multi-agent planning, context-awareness and adaptation are also very useful since they bring added value, by allowing to implement intelligent and collective behavior. The paper also presents a scenario of ambient intelligence and shows how it could be designed using the MAS paradigm.

1 Introduction to Multi-Agent Systems

A Multi-Agent System (MAS) [7] is an organization of a set of autonomous and potentially heterogeneous agents operating in a shared and dynamic environment. MAS represent (e.g. manage, model and / or simulate) physical systems (as is the case in the field of robotics) or (more often) software. The MAS keystone is the double inference mechanism that is used by the agents. Agents, unlike other design paradigms as objects or components, distinguish the level of task completion

Amal El Fallah Seghrouchni
LIP6, University Pierre et Marie Curie, 4 Place Jussieu, 75005 Paris, France
e-mail: amal.elfallah@lip6.fr

Adina Magda Florea
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania
e-mail: adina@cs.pub.ro

Andrei Olaru
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania (in cotutelle with LIP6, University Pierre et Marie Curie)
e-mail: cs@andreiolaru.ro

or problem solving from the level of control of the problem solving. Thus, they may act, observe their actions and change their own course of action. Agents have specific properties such as autonomy (an agent controls his condition and his actions regardless of any outside intervention); reactivity (an agent senses its environment and reacts to its changes); pro-activity (an agent tends to generate and achieve goals all by itself); and sociability (an agent interacts with other agents in the system).

Within a MAS, agents interact to achieve cooperative (e.g. distributed problem solving) or competitive (e.g. coalition formation, auction) group behavior. Finally, a MAS is deployed in a environment that impacts its dynamic behavior.

The dynamism of MAS covers several aspects:

- The dynamic structure of the MAS (the organization of agents) may change over time due to openness (arrival and departure of agents) and to the evolution of functional requirements (creation / removal of agents).
- The dynamics of acquaintances between agents: links can appear (arrival or creation of agents), others may disappear (departure or removal of agents) and / or change (e.g. for mobile agents).
- The dynamic environment of the MAS: changes in the environment are perceived by agents and taken into account incrementally.

At the agent level, it is the very structure of the agent that may change over time. Beyond changes in the cognitive elements of the agent (e.g. knowledge, goals, preferences, plans, etc.), we have proposed a model of agents whom are able to absorb other agents (their sub-agents) or conversely, to dissolve into their parents.

To capture these dynamics and concepts, we proposed an agent-oriented programming language that incorporates the cognitive foundations of MAS and extends them by means of local and distant mobility. Thus, a MAS is represented by a set of hierarchies distributed on multiple networked machines, each agent being a node in the hierarchy. It consists of cognitive elements (such as goals, beliefs), of processes, and of sub-agents.

An agent can dynamically acquire the knowledge, the capabilities and the sub-agents of sub-agents that it absorbs. The migration of agents, enriched with mechanisms of absorption and dissolution, allow the dynamic reconfiguration of MAS.

2 Context-Awareness

One of the central features that makes distributed systems ”intelligent” is *context awareness*. One of the definitions of context is that it is *the set of environmental states and settings that either determines an application’s behaviour or in which an application event occurs and is interesting to the user* [4]. One important point in the definition above is the relevance to the user. Either an event must be relevant to the user, or the application’s behaviour must change so that it becomes relevant to the user. Context awareness is that characteristic of an application that makes it change its behaviour in function of, and according to, context.

Research in the domain of context awareness has shown that there are many aspects of context. One classification of context [4] divides context into computational context – available computing and networking resources, including the cost for using them; user context – user's profile, location, people and objects nearby, social situation; physical context – light and noise levels, temperature, traffic conditions, etc; and time context – the current time coordinate of the user and related information (like the season, for instance). Context can be further classified [5] as primary – sensed directly by sensors and specialized devices – and secondary – which is inferred from the primary context.

If many authors consider context as merely a set of sensed values [1, 6], a particularly interesting approach to context-awareness is taken by Henricksen et al [8, 9], that model context as associations between entities or between entities and attributes, where an entity can be a person, a place, a communication device, etc. These associations can be of different types: static – associations that remain fixed for the lifetime of the entity; dynamic and sensed – obtained from sensors, usually transformed afterwards, changing frequently and subject to sensing errors; dynamic and derived – information that is inferred, usually from sensed or static associations; dynamic and profiled – introduced explicitly by the user, leading to greater reliability, but also subject to staleness.

In a context-aware system, there are several layers that deal with context information. One possible organization [12] uses three layers: data acquisition, data interpretation and data utilization. However, considering that much context information is volatile (e.g. user's location and time), a context-aware system must also feature components for the degradation of context information.

3 Mobile MAS Meets Ambient Intelligence

The agent-based paradigm is one of the paradigms that can be used for the implementation of distributed systems [7]. In the case of Ambient Intelligence, agents are particularly appropriate, because they offer features that originate from the field of Artificial Intelligence and that are vital to the needs of Ambient Intelligence [10]: proactive and reactive reasoning, autonomy, social abilities and learning. Autonomy is useful because individual devices in an Ambient Intelligence environment must be able to act on their own, without the need for user intervention or control from centralized components. Learning can serve to adapt to the user's habits. And reasoning – as well as the capability to make plans – is what makes a system appear intelligent to the user.

The agent-oriented paradigm is also useful in modeling real-world and social systems, where optimal solutions are not needed and problems are solved by cooperation and communication, in a fully distributed fashion [10]. Currently several agent-oriented programming languages exist [2], that allow the programmer to describe an application only by specifying the behaviour of individual agents.

Such an agent-oriented programming language is CLAIM, that also features a deployment platform for agents, called Sympa [11]. In CLAIM, each agent has a knowledge base, offers a to the exterior a certain number of capabilities and is capable both of reactive (by means of rules) and proactive behaviour. More importantly, the multi-agent system has a structure that is inspired from ambient calculus [3]: agents are placed in a hierarchical structure and an agent can have another agent as parent, as well as several other agents as children. Agents in CLAIM are mobile – they are able to change the host on which they are executing, and they are also able to change their place in the hierarchical structure. Moreover, when an agent moves, its children move with it automatically.

It is the hierarchical structure of the CLAIM multi-agent system that makes it especially appropriate for the implementation of an Ambient Intelligence system. That is because CLAIM makes it easier to implement context-awareness. An agent's ambient – formed by itself and all if its children can also represent a context. Agents can represent smart places, can manage smart devices, or can offer services.

4 A Case Study

Take for example the following scenario (also refer to Figure 1): a user has a meeting in a building that he / she does not previously know. When arriving at the right floor, the user's PDA automatically connects to a local wireless access point. A CLAIM agent executes on the user's PDA – we will call this agent *PDA*. Another agent executes on a local machine and manages the context of the building's floor – call it *Floor*. *Floor* detects the presence of the user's PDA, and instructs the *PDA* agent to move in the agent structure and become a child of *Floor*. The movement is only logical: the agents keep executing on the same machines as before.

When *PDA* enters the floor, *Floor* also spawns a new agent – called *Navigator* – and instructs it to move as a child of *PDA*. This time, the movement is not only logical: *Navigator* is a mobile agent that actually arrives on the user's PDA and will execute there for all the time during which the user is on the floor. The *Navigator* can provide *PDA* (and inherently the user) with a map of the floor, can translate indications of the floor's sensors (sent to *Navigator* by *Floor*, and through *PDA*)

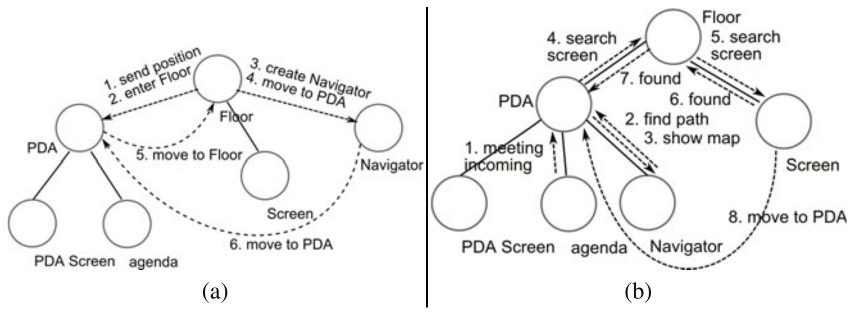


Fig. 1 Sequences of messages exchanged between agents: (a) *Floor* announces *PDA* of its new position, and instructs it to move as its child, then creates a *Navigator* that will offer services to *PDA*; (b) *Agenda* announces a new meeting, *PDA* asks a path from *Navigator*, which in turn asks for a larger screen – which is searched on the floor, and found, then *Screen* will move as a child of *PDA*.

into positions on the graphical map, and can calculate paths between the offices on the floor. *Navigator* is an agent that offers to the user services that are available and only makes sense in the context of the floor.

For displaying the map, *PDA* may detect that its screen is too small too appropriately display them map, so *PDA* will proactively initiate the search for a larger screen in the nearby area. The search can have several criteria: the space in which the search will take place (the current office, a nearby office, the whole floor), the range in which to search, and the minimal size of the searched screen. Devices are searched by the capabilities they offer – in this case the *display* capability is needed. *PDA* sends the query to its parent – *Floor* – which in turn locates among its children an agent *Screen*, that manages a physical screen that fits the requirements, is located near to the user and is available. *Screen* answers the query and *PDA* asks it to move to become its child. Being a child of *PDA* also marks the fact that *Screen* is in use by the user, and *PDA* gains control over the displayed information. Agent *Screen* may either run on the actual intelligent screen, or may only manage the screen while being executed on a server. When the user moves farther from the screen, the PDA will detect that the context is no longer compatible and will free *Screen*, which will return to be a child of *Floor*.

5 The Ao Dai Project

In the Ao Dai project, we have implemented, using CLAIM, a prototype of multi-agent system that handles several aspects of context-awareness, like user's location, available resources and user preferences. We have also implemented a scenario which is an extension of the one above. The project has been implemented by Thi Thuy Nga Nguyen, Diego Salomone Bruno and Andrei Olaru, under the supervision of Prof. Amal El Fallah Seghrouchni.

The prototype is implemented in CLAIM and executes on the Sympa platform. It features several types of agents: *Site*, which is used for "smart" places like *Floor* and *Office*; *PDA*, which directly assists the user from his personal device; *Navigator* and *Agenda*, which offer services to the user; and *Screen*, which represents a "smart" device with the capability of displaying information.

The prototype has been demonstrated during the 5th NII-LIP6 Workshop held on June 21-22 in Paris, France. The prototype was run on 2 machines. The *Floor* agent (of type *Site*) ran on one machine, and two *Office* agents (also of type *Site*) ran on the other machine. The floor and the two offices all featured screens of different sizes, managed by *Screen* agents (see Figure 2). During the demonstration, a *PDA* agent entered the floor, becoming a child of the *Floor* agent. A *Navigator* was created and sent to *PDA*. When the time of the meeting approached, *Agenda* announced *PDA*, which asked *Navigator* to find the path to the right office. *PDA* also searched for a larger screen, and found one near to the user, and automatically used it to display the map and the path. When the user – together with the PDA – moved to an office, the screen was freed and *PDA* with all children (*Agenda* and *Navigator*) moved to the other machine. There, the user explicitly requires a large screen, and *PDA* finds

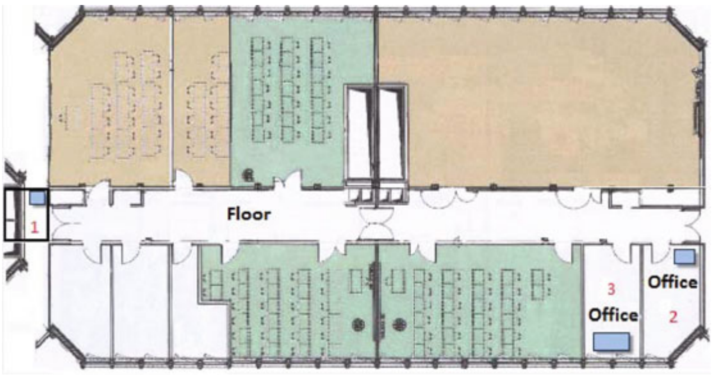


Fig. 2 The map shown by different screens in Ao Dai. There are three *Site* agents: *Floor* and two *Office* agents. Each one has a child of type *Screen*, representing the screens in the different places. The user starts on the floor (1) then moves to one office (2) and then to the other (3).

an appropriate one in the next room, and announces the user. The user then moves to the other office and *PDA* and its children all move to become children of the agent managing that office.

6 Conclusion

In this paper we showed how the MAS paradigm can be used to design intelligent and distributed systems. Hence, we have discussed some features of MAS such as natural distribution of MAS, inherent intelligence of the agents, and how mobile agents can help to address a large scope of applications in the domain of pervasive computing.

Other features of MAS like multi-agent planning, collective learning and adaptation deserve to be mentioned here. These additional features are very important since they bring added value by allowing intelligent collective behavior (the shift from single agent to multi-agent approach).

From our research perspective, the new challenges for MAS are related to their scalability and the balance to be found between the real time reaction of agents and the intelligent processing of the information gathered from the environment in this kind of applications. Another issue which is central, in particular in the context of pervasive computing, is the relation to users, which become a part of the system. Our future work at a short term is to provide a computational model to predict the users' intentions.

Acknowledgements. We would like to thank Thi Thuy Nga Nguyen and Diego Salomone Bruno for their work within Ao Dai project as Master degree training.

References

1. Baldauf, M., Dustdar, S., Rosenberg, F.: A survey on context-aware systems. International Journal of Ad Hoc and Ubiquitous Computing 2(4), 263–277 (2007)
2. Bordini, R.H., Braubach, L., Dastani, M., Fallah-Seghrouchni, A.E., Gómez-Sanz, J.J., Leite, J., O'Hare, G.M.P., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica (Slovenia) 30(1), 33–44 (2006)
3. Cardelli, L., Gordon, A.D.: Mobile ambients. Theor. Comput. Sci. 240(1), 177–213 (2000)
4. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College (2000)
5. Dey, A., Abowd, G.: Towards a better understanding of context and context-awareness. In: CHI 2000 workshop on the what, who, where, when, and how of context-awareness, pp. 304–307 (2000)
6. Feng, L., Apers, P.M.G., Jonker, W.: Towards context-aware data management for ambient intelligence. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 422–431. Springer, Heidelberg (2004)
7. Ferber, J.: Multi-agent systems: an introduction to distributed artificial intelligence. Addison-Wesley, Reading (1999)
8. Henricksen, K., Indulska, J.: Developing context-aware pervasive computing applications: Models and approach. Pervasive and Mobile Computing 2(1), 37–64 (2006)
9. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. LNCS, pp. 167–180. Springer, Heidelberg (2002), http://www.springerlink.com/content/jbxd2fd5ga045p8w/
10. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems 23(2), 15–18 (2008)
11. Suna, A., El Fallah Seghrouchni, A.: Programming mobile intelligent agents: An operational semantics. Web Intelligence and Agent Systems 5(1), 47–67 (2004)
12. Viterbo, J., Mazuel, L., Charif, Y., Endler, M., Sabouret, N., Breitman, K., El Fallah Seghrouchni, A., Briot, J.: Ambient intelligence: Management of distributed and heterogeneous context knowledge. CRC Studies in Informatics Series, pp. 1–44. Chapman & Hall, Boca Raton (2008)

Part II
Information Extraction and Web Applications

High-Level Web Data Abstraction Using Language Integrated Query

Jakub Misek and Filip Zavoral

Abstract. Web pages containing huge amount of information are designed for human readers; it makes their automatic computer processing difficult. Moreover web pages live their content is changing. Once a page is downloaded and processed, few seconds after that its content can be different. Many scraping frameworks and extraction mechanisms have been proposed and implemented; their common task is to download and extract required data. Nevertheless, the complexity of development of such application is enormous since the nature of data does not conform to common programming paradigms. Moreover, the changing content of the web pages often implies repetitive extracting of the whole data set.

This paper describes the LinqToWeb framework for web data extraction. It is designed in an innovative way that allows defining strongly typed object model transparently reflecting data on the living web. This mechanism provides access to raw web data in a completely object oriented way using modern techniques of Language Integrated Query (LINQ). Using this framework development of web-based applications such as data semantization tools is more efficient, type-safe, and the resulting product is easily maintainable and extendable.

1 Introduction

Since Internet lives in its own limitless world where everybody is allowed to join and contribute, all information placed on the web is growing every second. It is full of various data, like daily newspapers, discussion forums, shop catalogs, images or videos, which are accessible by almost anyone. The automatic data extraction represents huge nowadays problem. The web pages are intended for human readers. Moreover single information is spread on more pages. Although there exist recommended formats for web developers how to export their information to be

Jakub Misek · Filip Zavoral
Charles University in Prague, Czech Republic
e-mail: {misek,zavoral}@ksi.mff.cuni.cz

readable by machines (RSS, web services or more advanced and self-describing RDF or OWL), vast majority of the web data does not contain such semantic or structural description. The main goal of extraction tasks is to retrieve desired information into some structuralized form. That is the most common purpose of scraping frameworks and applications. They are able to download e.g. all needed web pages, and export specific values into a local storage like database or XML. Two main categories of scraping software approaches can be identified:

- **Standalone application.** Specific extraction mechanisms are implemented as a standalone large scale application [12, 13]. Such applications provide methods for extracting data from various sources and several options how to save the results. Everything used to be configurable in a declarative way, usually using a graphic user interface or scripting. Occasionally the application is able to perform updates automatically. These solutions have high requirements for storage capacity and the whole extraction process is time consuming.
- **Framework.** Programmers can take benefits from extraction frameworks [1, 8]. Using such libraries programmers have to take care about all the processes; the implying advantage is a possibility to develop more customized and optimized extraction. For example such application can modify extraction parameters during runtime or it can download only specific parts of web sites.

Most of these solutions assume that the user wants to extract information into a local storage. Such behavior is sometimes not desirable, because most of the downloaded information is not subsequently used. Downloading everything locally causes high requirements for storage capacity and difficult updates of already extracted information. Moreover many implementations solve the updates simply by redownloading everything from the beginning. The other issue when using such frameworks is efficiency of development data are usually represented in an unstructured form of strings, binding to language data types is weak, no syntax checking is possible in compile time etc.

The main contribution of this paper is a design of a high-level integration of web information extraction into a software development process. The framework LinqToWeb [11] benefits from advantages of previously described approaches. The extraction tasks can be defined in a declarative way, while the programming interface uses type safe object model representing the abstraction of the web resources. The tasks are compiled; it maximizes performance of the extraction.

The programmer can use web resources in the same way as data in a local memory. In contrast to contemporary solutions local storage is not used explicitly, but only as a transparent caching mechanism. Moreover, particular data items are accessed only when requested that makes the extraction process much efficient the data access is not delayed by long extraction queues. The high-level object oriented approach takes benefits of modern language features like Language Integrated Query (LINQ) [4], code sense capability provided by development environments etc. automatically.

2 Architecture

The main idea of the LinqToWeb design is working with web resources in the same way as with the statically declared objects instantiated during runtime. The object oriented approach takes advantage of safe type checking, efficient compiled task processing and auto-completion support provided by development environments - object members are automatically offered during code typing; the programmer can see the data structure immediately. Also the source code is automatically checked for typing errors; it makes the development easier and more type-safe.

The architecture is inspired by LinqToSQL [7] integration which was introduced as a part of Microsoft Visual Studio 2008. The main principle is based on generating strongly typed object model from something that is not so easy to use and causes programming issues in general. The description of data sources is used for generating type-safe objects that encapsulate all possible usages of data. The complete process of generating objects is automatically performed by the development environment - every time the source data description is changed, the programmer works with up-to-date objects implementation.

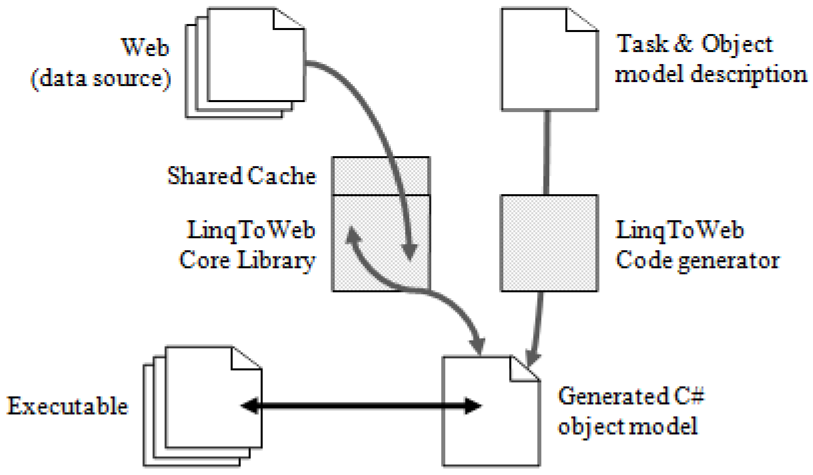


Fig. 1 LinqToWeb architecture. The programmer accesses only the generated object model, the rest is hidden under the generated abstraction.

The typical usage of LinqToWeb is depicted on Fig. 1. Assume we need to use information from specific web pages or another not necessarily web based sources. The programmer provides description of the information its abstract object model and how to fill this object with data (see Sect. 3). The LinqToWeb code generator transparently generates C# code containing objects and extraction tasks using functionalities hidden in LinqToWeb core libraries.

Generated code file becomes a part of the program sources; the compiler is able to perform type checks and optimizations. To achieve an effect of objects with all values locally initialized, the whole mechanism makes use of the .NET framework platform and its mechanism of properties. Moreover, generated objects can be integrated to all languages supporting the .NET platform. During the extraction the system uses associative cache. Its purpose is to speed up repetitious requests for web resources or repetitious extraction tasks. This cache manages expiration time of resources; expired cached resource is simply downloaded again as part of evaluation of next relevant request. The storage of the cache can be implemented in various ways, e.g. in memory only or in a database. Sharing the cache among more threads or processes simplifies concurrent access to a particular object.

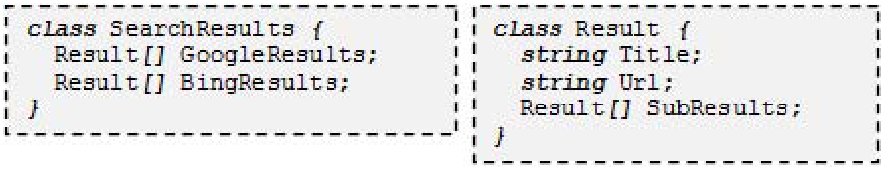
3 LinqToWeb Description Language

The description of each data source consists of two parts - the structure of information and the extraction tasks that collect it. A special-purpose declarative language was designed [10] to describe both parts. Its main features include: simplicity, declarative task description, procedural processing logic, intuitive object oriented interface, and support for inherent and transparent parallelism.

3.1 Type Declarations

The LinqToWeb language offers built-in frequently used types and the ability to define user-defined classes. The type system is divided into value types and extraction object types. Built-in value types are: *int*, *double*, *datetime*, *bool* and *string*. The extraction objects are passed by reference through the extraction tasks. Therefore they can be filled with data subsequently and used as a result of extraction. They are represented by built-in container list and all user-defined classes.

Fig. 2 demonstrates the declaration of user-defined classes containing both list (denoted by *[]*) and built-in properties, they represent the abstraction that the programmer work with.



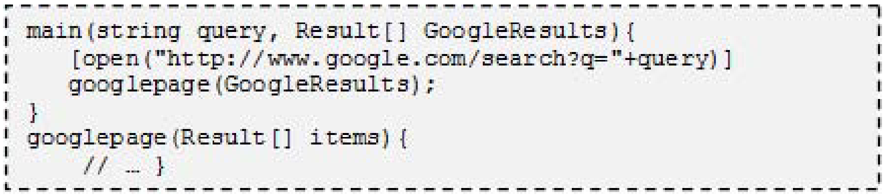
```
class SearchResults {
  Result[] GoogleResults;
  Result[] BingResults;
}
```

```
class Result {
  string Title;
  string Url;
  Result[] SubResults;
}
```

Fig. 2 Example of structuralized type declaration, classes and lists

3.2 Extraction Task Definition

The process of obtaining data is represented by extraction methods. Every method is defined by a method name, its typed arguments and a body. The arguments are used to pass values and references to objects that the method will use or fill in. Arguments of value types are passed by value and arguments of other types are passed by reference. Fig. 3 demonstrates the extraction methods definition. The entry point of extraction is represented by the method main. Value-typed arguments of these methods are automatically exposed as arguments of the extraction, arguments typed as extraction objects are exposed as public objects containing extraction results.



```
main(string query, Result[] GoogleResults){
   [open("http://www.google.com/search?q="+query)]
   googlepage(GoogleResults);
}
googlepage(Result[] items){
    // … }
```

Fig. 3 Extraction methods definition example. Calls a method *googlepage* on given web page.

Body of extraction method. Method body consists of statements and expressions. The language supports common literals and common operations like numeric operators, strings concatenation and assignments. The code looks like a single-threaded extraction process, but the real execution uses the code that is generated from this declaratively defined task which may process the requests in a parallel and optimized way. Hence it allows the programmer to define the abstraction without concerning e.g. any parallelization issues.

Extraction method invocation. The calling syntax is as usual as in other procedural languages. But there is a difference in the semantics. Methods represent the smallest tasks which execution can be postponed. The code looks like a single process but in fact method calls are not processed immediately the call is postponed until the data is requested. Within the method call, only the method arguments remember that their content can be affected by the called method. Therefore the method is really processed only if the content of the required object is unknown yet.

The language also allows definition of more methods with the same identifier and signature (multimethods). When multimethods are invoked, all their instances are called in an undefined order; such behavior is especially useful for exploiting parallelism and other optimizations.

Data contexts. Since the compound result set can be extracted from various sources, the language introduces data-contexts. Every statement is executed within a data-context which represents current open resource, e.g. current web page. Syntactically, new data-context can be open using square brackets before the statement, including command that opens the resource. There is the open command including location of the web page in Fig. 3, 5. The location is relative to the current data-context; opening a resource simulates e.g. following a link from the current page.

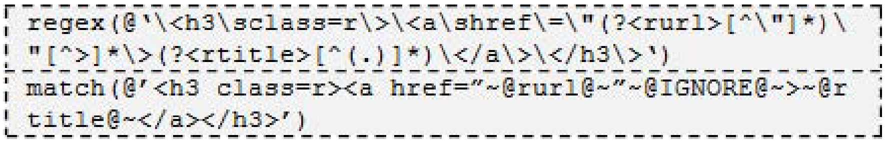
The purpose of data-contexts is reuse of the code with different source data. They also simplify design of the extraction tasks. It allows the programmer to write the code more declaratively in a similar way as the human browses the web pages.

Other language constructs. Several other constructs are needed to provide fully usable language. Although most of them are known from procedural languages, there are some differences. Following list describes the most important features:

- **Item addition.** The list object is used as container for elements. In extraction methods lists are write-only. To add an element into the list the following syntax is used: `items[]=element`; In the program that uses generated code, the lists are read-only; they are commonly used as data pipes.
- **Object creation.** To create and initialize an object within a single expression the syntax similar to a method call is used. It is useful when new object has to be created, its properties initialized and the result object added to a list or passed as a method call argument. The syntax looks as follows: Result(Title=..,Url=..).
- **.NET code call.** Sometimes it is useful to use standard C# code. The system supports such methods; they can be used and called in the same way as LinqToWeb methods, but without a possibility of delayed processing.

Extractor patterns. Easy data extraction is the main purpose of the extraction methods. The example of collecting results from the Google search is shown on Fig. 5. All required fragments of a source web page are enumerated; data are extracted from the current data-context. The argument of the `foreach` specifies the searched extractor pattern. When a match is found the system automatically creates local variables and initializes them with the matched data. The system currently supports following extractor patterns (particular examples are shown on Fig. 4, 5):

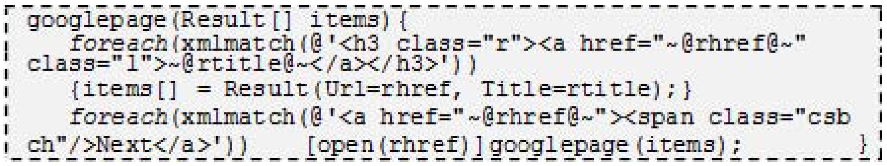
- **Regexp.** Matching the regular expression [6] is the basic and most commonly used method of navigating the web page. However this method is very hard to use



```
regex(@'\<h3\sclass=r\>\<a\shref\=\"(?<rurl>[^\"]*)\
"[^>]*\>(?<rtitle>[^(.)]*)\</a\>\</h3\>')
```

```
match(@'<h3 class=r><a href="~@rurl@~"~@IGNORE@~>~@r
title@~</a></h3>')
```

Fig. 4 Examples and comparasion of *Regexp* and *Mat Extractor Pattern*



```
googlepage(Result[] items){
    foreach(xmlmatch(@'<h3 class="r"><a href="~@rhref@~"
class="l">~@rtitle@~</a></h3>'))
    {items[] = Result(Url=rhref, Title=rtitle);}
    foreach(xmlmatch(@'<a href="~@rhref@~"><span class="csb
ch"/>Next</a>'))    [open(rhref)]googlepage(items);        }
```

Fig. 5 Extraction method example, using *XML Extractor Patterns*

in case of complex expressions. The variables in the expression can be identified in the following way: `(?<var_name>_expression_)`.

- **Match.** Instead of regular expressions, *Mat Extractor Patterns* can be used. They were introduced in scraping framework AgentMat [3]. *Mat Extractor Patterns* are much more easy to use, but they are not as strong as *regexp*. They consist of piece of text that will be matched. This text is converted into corresponding regular expression. The variables are marked by prefix `~@` and suffix `@~`.
- **Xmlmatch.** LinqToWeb system introduces *XML Extractor Patterns*. They have a form of a fragment of valid XML. The content of the data-context is parsed into XML DOM. *XmlMatch* searches for all matches against the provided XML fragment. Variables are marked in the same way as it is in the *Mat Extractor Patterns*, but they can occur only in the XML attributes value and inner nodes text. *XML Extractor Patterns* allows finding matches within the context of HTML nodes easily. It is also more tolerant for changes in the source web page structure, the programmer is not forced to describe elements and attributes of the source HTML text that are not important for the match, only the structure must be specified. In this case, matched HTML fragment can contain other XML attributes or even other XML child elements. The XML pattern is well-arranged with a possibility to specify wider context of searched information on a higher level of declaration.

4 Generating Strongly Typed Objects

The tasks described in the previous section are used to generate objects in C# language. The result is in a form of a context class that contains all exported objects declared and instantiated inside.

The context class has public read-only properties corresponding to exported objects. They are defined as arguments of the *main* extraction methods typed as extraction objects. Declarations of user-defined classes are placed within the context class too. Finally the constructor of the context class is parameterized. The parameters correspond to value-typed arguments of the *main* method.

4.1 Just-In-Time Extraction

Every non-value-typed object is derived from an abstract class which defines the mechanism of actions. It manages a collection of actions that lead to collecting data for this object. Every action consists of a method to be called and its parameters. When a property of an extraction object is to be read, the getter method of the property processes and removes some action if the value is not known yet. New actions appear in the collection by calling extraction methods as it was mentioned before and by a context class constructor.

By processing the action, some unknown information can be obtained and stored in object properties. Also new actions can be added into the collection. The getter method of the requested property processes actions until the property gets a value.

If the particular property is read repeatedly, its value is extracted only once. If the property is not read, its extraction is not performed at all.

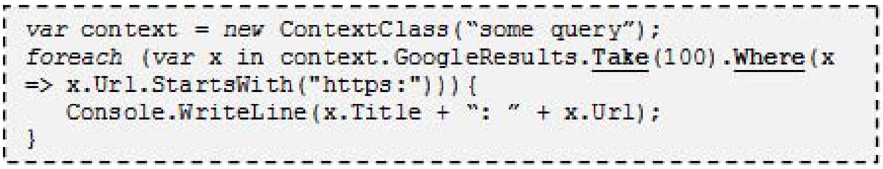
Enumerating lists. Typically, the main result of the extraction task is a set of collections of data. The collections are represented by the list-typed objects. It is also derived from the abstract base class; therefore it manages the mechanism of actions. Instead of using properties, the list object is targeted to act like an enumerator [9] (or a read-only pipe in other words); it can be used only for getting items sequentially.

The runtime uses this enumerator automatically when the program is accessing the collection. The actions associated with the collection are used when the runtime picks up next element from the enumerator, next action from the actions list is processed to obtain more elements if necessary.

Every time the program enumerates the list new enumerator is created and list elements are obtained from beginning. Since frequently used resources and results of extraction operations are cached, such behavior does not affect the performance. Moreover it allows defining endless collections without requirements for endless storage capacity. Also only the beginning of the collection is extracted.

4.2 *Language Integrated Query*

The system of properties and enumerators allows taking benefits from LINQ easily. It automatically extends all the enumerator objects in .NET with other methods. They implement common operations over collections like *Sort*, *Count*, *GroupBy*, *Sum*, *TakeWhile* etc. As the query is processed all the missing information (e.g. value of properties or enumerated items) are collected and cached transparently.



```
var context = new ContextClass(“some query”);
foreach (var x in context.GoogleResults.Take(100).Where(x
=> x.Url.StartsWith("https:"))){
   Console.WriteLine(x.Title + “: ” + x.Url);
}
```

Fig. 6 Enumerating lists and LINQ usage in C#. Advanced queries on pipes representing web collection. Writing the single elements onto the output in object oriented and type safe way.

Fig. 6 demonstrates the usage of lists and LINQ extensions. The sample collects results from the Google search; it takes first 100 results and selects only items which URL uses secured HTTP protocol. The programmer can work with collections extracted from the web in a completely abstract object oriented and type safe way. He has also access to more advanced queries without a need of implementing them.

5 Evaluation

Using the LinqToWeb framework several applications have been developed, especially for web semantization subprojects [2, 5]. The code fragments referenced from previous sections illustrate use of the language. The object model is generated from Google search pages; this model is subsequently used to fetch and process the data.

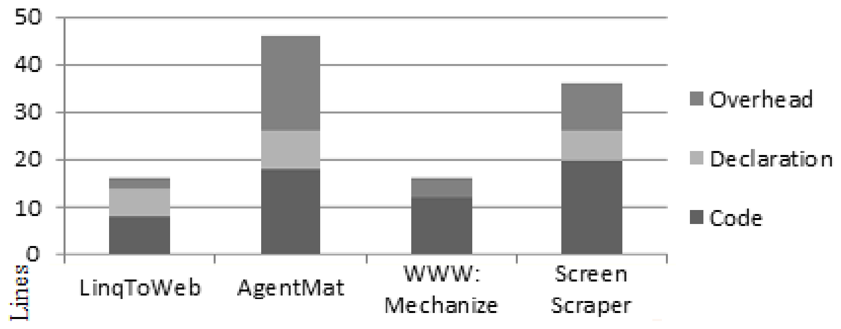


Fig. 7 Extraction frameworks and written code complexity

The size of the code needed to use Google results in the application is depicted on Fig. 7. Most of the frameworks are able to define the task in a declarative way, but the usage of extracted data is not trivial (*AgentMat* and *ScreenScraper* systems). Smaller code means faster development and less chance for errors. Also other frameworks used as libraries force the programmer to mix extraction task within the program itself (e.g., *WWW: Mechanize*). The graph shows amount of code lines defining the extraction task and its usage, declaring object structure or tasks and also the syntax overhead. The overhead is needed to proper usage of the framework, but it does not affect the extraction itself.

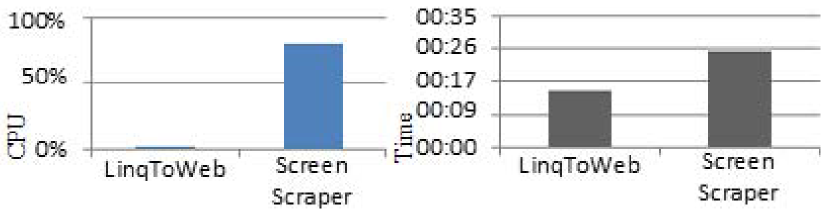


Fig. 8 CPU usage and extraction task speed, compiled vs. interpreted tasks

The LinqToWeb extraction tasks are compiled; it results in faster processing and lower CPU usage, the process is waiting for web responses most of the time. The difference of compiled and interpreted task is depicted on Fig. 8. Both frameworks were used to obtain 400 results from Google search on Core2Duo CPU and a standard broadband connection.

6 Conclusion and Future Work

This paper describes methods used for integrating information from web-based sources into application development. Implemented framework LinqToWeb enables reading data in a type safe and object oriented way. The methods are suitable for frequently updated live data. Using generated object model, data are accessed as strongly typed objects. Hence it allows taking benefits of compiler type checks and IDE code sense capabilities. Moreover, downloading, updating and data extraction is performed on request fully transparently.

The main future enhancement is intelligent ordering and automatic parallelization of processed extraction tasks. Other possible enhancements are related to using a semantic storage for caching. Also some LINQ methods can be extended to modify the extraction process by using the information from the query, e.g. data can be downloaded already sorted if the source web allows it. Other kinds of improvements include automatic detection of a web page structure and its semantic content. Since the framework is intensively used in a web semantization research project, we expect its future development in other not mentioned aspects.

Acknowledgements. This work was supported in part by grants GACR 202/10/0761, SVV-2010-261312, and GAUK 2010/28910.

References

1. Gisle Aas: HTML Parser informations, http://search.cpan.org/~GAAS/HTML-Parser/
2. Bednarek, D., Dokulil, J., Yaghob, J., Zavoral, F.: Using Methods of Paral-lel Semi-structured Data Processing for SemanticWeb. In: Proceedings of SEMAPRO 2009. IEEE Computer Society Press, Los Alamitos (2009)
3. Beno, M., Misek, J., Zavoral, F.: AgentMat: Framework for Data Scraping and Semantization. In: RCIS, Fez, Morocco (2009)
4. Box, D., Hejlsberg, A.: LINQ:NET Language-Integrated Query. In: MSDN (2007)
5. Dokulil, J., Yaghob, J., Zavoral, F.: Trisolda: The Environment for Semantic Data Processing. International Journal On Advances in Software 2008, IARIA 1(1) (2009)
6. Friedl, J.: Mastering Regular Expressions. O'Reilly Media, Inc., Sebastopol (2006)
7. Kulkarni, D., Bolognese, L., Warren, M., Hejlsberg, A., George, K.: LINQ to SQL:NET Language-Integrated Query for Relational Data
8. Lester, A.: WWW:Mechanize, http://search.cpan.org/~petdance/WWW-Mechanize-1.52/
9. Mackay, C.A.: Using .NET Enumerators, The Code Project (2003), http://www.codeproject.com/KB/cs/csenumerators.aspx
10. Misek, J.: LinqToWeb Language Definition, Technical report KSI 2010/01, Charles University in Prague (2010)
11. Misek, J.: LINQ to Web project, http://linqtoweb.codeplex.com/
12. Ekiwi: Screen scraper informations, http://www.screen-scraper.com/
13. Kapow Technologies: Kapowtech Mashup Server informations, http://www.kapowtech.com

Semantic Web Service Clustering for Efficient Discovery Using an Ant-Based Method

Cristina Bianca Pop, Viorica Rozina Chifu, Ioan Salomie, Mihaela Dinsoreanu, Tudor David, and Vlad Acretoaie

Abstract. This paper presents an ant-inspired method for clustering semantic Web services. The method considers the degree of semantic similarity between services as the main clustering criterion. To measure the semantic similarity between two services we propose a matching method and a set of metrics. The proposed metrics evaluate the degree of match between the ontology concepts describing two services. We have tested the ant-inspired clustering method on the SAWSDL-TC benchmark and we have evaluated its performance using the Dunn Index, the Intra-Cluster Variance metric and an original metric we introduce in this paper.

1 Introduction

Web service discovery has been widely discussed in the area of distributed systems for the past few years. The prospects of discovering services have considerably improved with the appearance of semantic Web services. However, the existence of a large number of available services makes the discovery process inefficient, unless services are grouped according to specific criteria. Service clustering is a solution to this problem as it aims to gather services sharing common semantic features into clusters.

This paper proposes a new approach to service clustering inspired by the behavior of ants. Our goal is to organize large sets of Web services into clusters based on the similarity between their semantic descriptions. A semantic service description is related to a domain ontology which includes concepts used to annotate the service input and output parameters. To evaluate the semantic similarity between two services we define a set of metrics that consider the ontology hierarchies of concepts and properties.

Cristina Bianca Pop · Viorica Rozina Chifu · Ioan Salomie · Mihaela Dinsoreanu · Tudor David · Vlad Acretoaie
Department of Computer Science, Technical University of Cluj-Napoca, 26-28 Baritiu str., Cluj-Napoca, Romania
e-mail: {Cristina.Pop,Viorica.Chifu,Ioan.Salomie}@cs.utcluj.ro

The paper is structured as follows. Section 2 presents related work. Section 3 presents our approach for evaluating the semantic similarity between two services, while section 4 illustrates the proposed ant-based service clustering method. Section 5 highlights the experimental results and their evaluation. We end our paper with conclusions.

2 Related Work

The problem of data clustering can be approached by using the classical partitioning and hierarchical clustering techniques. A comparison between the partitioning and hierarchical techniques is presented in [1] and can be summarized as follows: partitioning clustering as opposed to hierarchical clustering avoids the problem of overlapping clusters, while hierarchical clustering as opposed to partitioning clustering does not require to know in advance the number of clusters to be discovered. Two methods for service clustering using a classical technique are presented in [9] and [8]. The method proposed in [9] relies on the hierarchical agglomerative technique to identify service clusters based on a given query and by considering the services' syntactic descriptions as clustering criteria. On the other hand, the method proposed in [8] applies the hierarchical agglomerative technique to cluster services based on their semantic descriptions.

Besides the classical techniques, researchers consider the use of biologically-inspired methods for clustering [5] [11] [1] [2] [7]. In what follows we present a comparative analysis as reported in the literature between the classical and biologically-inspired techniques for clustering. In [1] it is stated that the clustering method based on Particle Swarm Optimization is better than partitioning clustering as it avoids the problem of local optima stagnation. The advantages of ant-based clustering compared to the classical clustering methods are presented in [5]. According to [5], ant-based clustering does not require to know in advance the number of clusters and the obtained clusters have a higher quality. Research literature reports a classical [5] and a hybrid [11] ant-based clustering technique. The hybrid technique, Tree Traversing Ant (TTA), combines features of ant-based clustering with features of classical clustering techniques. In the case of service clustering, we have identified in [6] a method based on TTA that considers the services' syntactic descriptions as clustering criteria.

In this paper we propose a semantic service clustering method which adapts the ant-based clustering algorithm presented in [5]. Our method clusters the services based on their semantic descriptions, as opposed to [6] and [9] which consider only the syntactic descriptions. By considering the semantic descriptions, the clustering process is improved, as it enables reasoning which leads to refined results. By applying an ant-inspired algorithm for service clustering it is not neccessary to specify in advance the number of clusters to be created and also the problem of overlapping clusters is avoided. In addition, to refine the clustering results we employ a method that allows the isolated services to migrate to the most suitable cluster.

3 Evaluating the Semantic Similarity between Services

A prerequisite for the ant-based service clustering method is the evaluation of the degree of semantic similarity between two services. We consider that two services are similar if there is a high degree of match (*DoM*) between their semantic descriptions (*DoM* is closely to 1, where *DoM* $\in$ [0, 1]). In our approach, a semantic service description consists of ontology concepts annotating the service input/output parameters.

To evaluate the *DoM* $(DoM_{services})$ between two services we adopt a decomposition technique. First, we compute the $DoM_{inputs}/DoM_{outputs}$ between the concepts describing the input/output parameters of the services. Then we combine DoM_{inputs} and $DoM_{outputs}$ to obtain a value that reflects the *DoM* between the two services.

1. Evaluation of DoM_{inputs} **and** $DoM_{outputs}$. The evaluation process of DoM_{inputs} and $DoM_{outputs}$ is the same and implies the evaluation of *DoM*s between pairs of concepts annotating the input/output service parameters and combining the results.

1.1 Compute $DoM_f(c_1, c_2)$ **between a pair of concepts**. DoM_f (formula 5) has two components: one representing the match between the concepts themselves and one representing the match between their properties. The properties of the concepts are considered to refine the process of evaluating the degree of match between a pair of concepts. The steps for computing DoM_f are presented below:

1.1.1 *Compute* $DoM(c_1, c_2)$ *between two concepts based on the concepts hierarchy*. To compute $DoM(c_1, c_2)$ we adapt the formula in [10] as follows:

$$DoM(c_1, c_2) = \frac{|\{c|c \subseteq c_1\}|}{|\{c|c \subseteq nca(c_1, c_2)\}|} * \frac{|\{c|c \subseteq c_2\}|}{|\{c|c \subseteq nca(c_1, c_2)\}|} \tag{1}$$

In formula 1, the inclusion operator denotes the sub-concept relation and *nca* is the nearest common ancestor of c_1 and c_2 in the considered ontology. We consider that there is a non-zero matching value between c_1 and c_2 even if they are siblings.

1.1.2 *Compute DoM(P(c_1), P(c_2)) between two concepts based on the properties hierarchy*. For computing *DoM(P(c_1), P(c_2))* we define the formula:

$$DoM(P(c_1), P(c_2)) = \frac{DoM_p(P(c_1), P(c_2)) + DoM_p(P(c_2), P(c_1))}{2} \tag{2}$$

where *P(c_1), P(c_2)* represent the sets of properties associated to concepts c_1, c_2 and $DoM_p(P(c_1), P(c_2))$ is the degree of match between the two sets of properties. The $DoM_p(P(c_a), P(c_b))$ is computed as an average matching value between the properties of concepts c_a and c_b related to all properties of concept c_a:

$$DoM_p(P(c_a), P(c_b)) = \frac{\sum_{i=1}^{|P(c_a)|} max_{j=1}^{|P(c_b)|} DoM(p_i(c_a), p_j(c_b))}{|P(c_a)|} \tag{3}$$

In formula 3, $p_i(c_a)$, $p_j(c_b)$ denote the i^{th}, j^{th} property of concepts c_a, c_b and $DoM(p_i(c_a), p_j(c_b))$ is the *DoM* between two properties $p_i \in P(c_a)$ and $p_j \in P(c_b)$. Formula 3 is asymmetric because the values for $DoM_p(P(c_1), P(c_2))$ and

$DoM_p(P(c_2),(P(c_1)))$ are not equal if c_1 and c_2 have a different number of properties. As a result, we decided to compute the *DoM* between $P(c_1)$ and $P(c_2)$ as well as between $P(c_2)$ and $P(c_1)$ to obtain a symmetrical matching (formula 2). The *DoM* between two properties $p_i \in P(c_1)$ and $p_j \in P(c_2)$ is computed as:

$$DoM(p_i,p_j) = \frac{|\{p|p \subseteq p_i\}|}{|\{p|p \subseteq nca(p_i,p_j)\}|} * \frac{|\{p|p \subseteq p_j\}|}{|\{p|p \subseteq nca(p_i,p_j)\}|} \tag{4}$$

1.1.3 *Compute $DoM_f(c_1,c_2)$ between two concepts based on the concepts and properties hierarchy*. To compute DoM_f we combine formulas 1 and 2 in formula 5 using an arithmetic mean. We use the arithmetic mean because it provides numerically bigger results which widen the range of possible *DoM* values and ensure a better basis of forming clusters.

$$DoM_f(c_1,c_2) = \frac{DoM(c_1,c_2) + DoM(P(c_1),P(c_2))}{2} \tag{5}$$

1.2 Compute DoM_{inputs} / $DoM_{outputs}$ between two sets of concepts annotating the inputs/outputs of two service descriptions. The DoM_{inputs} between the sets of input parameters of services s_1 and s_2 is defined as follows:

$$DoM_{inputs}(s_1,s_2) = DoM_{in}(s_1,s_2) * DoM_{in}(s_2,s_1) \tag{6}$$

DoM_{in} in formula 6 is computed as:

$$DoM_{in}(s_a,s_b) = \frac{\sum_{i=1}^{|in(s_a)|} max_{j=1}^{|in(s_b)|}(DoM_f(c_a,c_b))}{|in(s_a)|} \tag{7}$$

where $in(s_a)$, $in(s_b)$ are the set of concepts annotating the input parameters of services s_a, s_b and c_a, c_b are two concepts annotating an input of s_a, s_b. The $DoM_{outputs}$ between the outputs of the service s_1 and the outputs of the service s_2 is computed with the same formulas as in the case of input parameters (formulas 6, 7).

2. Evaluation of $DoM_{services}$. The final step in the service matching process is to combine the DoM_{inputs} and $DoM_{outputs}$ into a single value, $DoM_{services}$. This result will be used as a measure of similarity between services in the clustering process. $DoM_{services}$ is obtained according to formula 8 as an arithmetical mean:

$$DoM_{services}(s_1,s_2) = \frac{DoM_{inputs}(s_1,s_2) + DoM_{outputs}(s_1,s_2)}{2} \tag{8}$$

4 The Ant-Inspired Solution for Clustering Services

For grouping similar services into the same cluster we have adapted the ant-based algorithm described in [5]. We establish whether two services are similar by using the metrics proposed in section 3. We also impose the restriction that the number of inputs of two services belonging to the same cluster should be the same or should differ by at most 1.

4.1 Problem Formalization

The clustering phenomenon can be observed at ants in the process of constructing cemeteries [3]. If a large number of corpses are present in the vicinity of a colony, ants will start picking them up, move them for a while, and then drop them. In time, small heaps will start to form which will merge, resulting in larger and larger heaps. Just as ants pick up and move corpses, we have a number of artificial ants which pick up and move services within our artificial environment. Thus, the ants clustering behavior is mapped to the service clustering as follows: (*i*) an ant becomes an artificial ant; (*ii*) the environment in which the ant moves becomes a toroidal grid [5]; (*iii*) a corpse the ant moves becomes a service; and (*iv*) a cluster of corpses the ant builds becomes a cluster of similar services. In our approach, the toroidal grid is represented as $GRID = [grid_{ij}]_{i=1,n;j=1,m}$ where an element of $GRID$ is defined as:

$$grid_{ij} = \begin{cases} id_s, & \text{if there is a service on row } i \text{ and column } j \text{ in the grid} \\ -1, & \text{otherwise.} \end{cases} \tag{9}$$

where id_s is an identification number associated to a semantic Web service (formula 10). We formally define a Web service as:

$$s = (id_s, i, j, id_c, in, out) \tag{10}$$

where: (*i*) id_s is an identification number associated to the service s; (*ii*) i, j represent the row and the column of the grid element where the service s is positioned; (*iii*) id_c is an identification number of the cluster to which the service s belongs; (*iv*) in and out are two sets of ontology concepts annotating the input/output parameters of s. We denote with S the set of available services to be clustered.

Just as ants gather corpses based on their size, artificial ants should create service clusters containing similar Web services. An artificial ant is defined as:

$$ant = (i, j, memory, s) \tag{11}$$

where: (*i*) i, j represent the row and the column of the grid element where the ant is positioned; (*ii*) memory keeps a history of the last k elements of the $GRID$ that were visited by the ant; (*iii*) s is a service the ant carries (will be dropped in the $GRID$).

We define an ant cluster in the context of semantic Web services as:

$$cl = (id_c, S_c) \tag{12}$$

where id_c is the cluster's identification number. We denote with CLS the set of all service clusters.

4.2 The Clustering Algorithm

Our ant-based service clustering algorithm (**Algorithm 1**) groups similar services into clusters in order to reduce the search space of the service discovery process.

Algorithm 1. Ant_Service_Clustering

```
Input: $S$ - set of services to cluster; $noAnts$ - number of artificial ants; $CLS$ - set of previous clusters;
Output: $CLS$ - the updated set of clusters;
Comments: $SP$ - set of service partitions; $sNo$ - the number of previously processed services;
begin
  if $((sNo > \Delta * |CLS|)$ or $(CLS == \emptyset))$ then
      if $(CLS == \emptyset)$ then $CLS = S$
      $SP =$ Partition$(CLS)$; $CLS = \emptyset$
      forall $sp_i \in SP$ do $CLS = CLS \cup$ Ant_Clustering$(sp_i, noAnts)$
      $sNo = 0$
  else
      forall $s \in S$ do
          $CLS =$ Add_Service$(CLS, s)$
          $sNo = sNo + 1$
      end for
  end if
  return $CLS$
end
```

The clustering algorithm is applied in two cases: when it creates clusters of services from scratch and when it distributes a set of services to an existing set of clusters *CLS*. The first case appears when the condition $sNo > \Delta * |CLS|$ holds (a number *sNo* of services are distributed among the clusters in *CLS* and the reclustering of the whole *CLS* should be performed) or when *CLS* is empty. The following steps are performed in the first case: (1) the services in *CLS* are partitioned (**Partiton**) according to the number of inputs and the partitions are stored in *SP*; (2) *CLS* is emptied; (3) each service partition is submitted to the ant clustering algorithm (**Ant_Clustering**) and the resulting clusters are added to *CLS*; (4) *sNo* becomes 0. For the second case, the condition $sNo > \Delta * |\text{CLS}|$ does not hold thus the services given as input just need to be distributed among the clusters in *CLS* (**Add_Service**).

Algorithm 2 clusters a given set of services. The services are first randomly placed on the *GRID* (**Initialize_Grid_Of_Services**) and the artificial ants are randomly associated to a service and to a location (**Pick_Service_Location**). The artificial ants start moving services around for a fixed number of iterations. In each iteration an artificial ant moves in the grid (**Move**) and decides whether to place the service in a grid element (**Decide_Drop**). The first move is influenced by the best position found so far kept in the ant memory. If the artificial ant does not decide to drop the service in this best position, then the ant performs random moves.

The decision to pick up (**Decide_Pick**) or drop (**Decide_Drop**) a service is probabilistic and takes into account the *DoM* between the service and the surrounding

Algorithm 2. Ant_Clustering

```
Input: $S$ - the set of services; $noAnts$ - the number of artificial ants;
Output: $CLS$ - the set of clusters;
Comments: $G$ - the toroidal grid; $noIt$ - the number of iterations; $Ants$ - the set of artificial ants;
begin
  $Ants = $ Initialize_Ants($noAnts$); $G = $ Initialize_Grid_Of_Services($S$)
  forall $ant \in Ants$ do $ant = $ Pick_Service_Location($S, G$)
  for $i = 0$ to $noIt$ do
      $ant = $ Choose_Random_Ant($Ants$)
      $ant = $ Move($G$)
      if (Decide_Drop($ant$)) then
          $ant = $ Drop_Service($G$)
          while (Decide_Pick($ant$) == false) do $ant = $ Move($G$)
          $ant = $ Pick_Service($S$)
      else $ant = $ Move($G$)
      end if
  end for
  $CLS' = $ Retrieve_Clusters($S, G$); $CLS = $ Clustering_Refinement($CLS'$)
  return $CLS$
end
```

services on the grid. The decision to drop a service is an implementation of formula 13, while the decision to pick a service is based on formula 17. The only adaptations made to these procedures for dealing with semantic Web services was to define the distance between two services as inveresely proportional to the *DoM* between the two services. The decision of placing a service in a grid element is taken according to a probability P_{drop} defined in [5] as:

$$P_{drop}(ant, ant.s) = min(1.0, f^4) \tag{13}$$

Formula 14 adapts the f defined in [5] to the case of semantic Web services:

$$f = \begin{cases} \frac{1}{N} * \sum_{j \in L_S} (1 - \frac{\Delta(ant.s, s_j)}{\alpha}), \text{ if } (1\text{-}\frac{\Delta(ant.s,s_j)}{\alpha})>0 \\ 0, \text{otherwise.} \end{cases} \tag{14}$$

where: (i) N normalizes f and depends on the neighborhood size and the number of services in that neighborhood. (ii) Δ evaluates the *DoM* between two services s_i, s_j and is computed as:

$$\Delta(s_i, s_j) = 1 - \sqrt{DoM_{services}(s_i, s_j)} \tag{15}$$

(iii) L_s is the set of services in the neighborhood of the ant position:

$$L_s = \{s_l \mid \forall s_l, |s.i - s_l.i| < \frac{\delta}{2} \; and \; |s.j - s_l.j| < \frac{\delta}{2}\} \tag{16}$$

In formula 16, δ is a value equal to the size of the neighborhood which increases linearly during the run of the clustering algorithm. If the ant decides to drop the service (**Decide_Drop**), it places the service in a grid element (**Drop_Service**) and then moves in the grid (**Move**) until it decides to pick-up another service (**Decide_Pick**). Just as in the case of dropping a service, the decision to pick-up a service is taken with a probability P_{pick} [5], where f is defined as in formula 14:

$$P_{pick}(ant, s) = min(1.0, \frac{1}{f^2}); \tag{17}$$

At the end of Algorithm 2, once similar services are within a neighborhood, a cluster retrieval algorithm (**Retrieve_Clusters**) is run, which identifies the neighborhood of services and organizes them into clusters. The neighborhood is established using a cluster retrieval distance between two services. As a final step of the clustering algorithm, a clustering refinement procedure (**Clustering_Refinement**) is executed. This procedure finds the remaining isolated services and checks if they fit into one of the already formed clusters. If they belong to one of these clusters (the average *DoM* between the isolated service and the services already in the cluster is above a certain value), they are attached to it. Thus, clusters containing a single service are eliminated, remaining only if a service does not match with any service.

The **Add_Service** procedure in Algorithm 1 addresses the case in which a service *s* is added to the existing set of clusters *CLS* without re-clustering the whole *CLS*. This procedure first places the service *s* in a randomly chosen free element of the grid and also, an ant is positioned in that place. For a given number of iterations, the ant just picks and drops the service *s* until it is placed in a suitable cluster.

5 Results and Evaluation

We have tested our ant-based service clustering method on SAWSDL-TC [12], a benchmark containing 894 services, proposed for evaluating the performances of service matchmaking algorithms. By analyzing the experimental results we noticed that there are several parameters which influence the behavior of the clustering algorithm, two of them being the Cluster Retrieval Distance (CRD) and the Number of Iterations (NoIt) the clustering algorithm performs. In our experiments we have considered a recommended [5] minimum number of iterations *NoIt* = (2000 * the number of services to be clustered). We performed a set of experiments on the SAWSDL-TC benchmark by varying the values of CRD and NoIt (see table 1).

We evaluated the experimental results with the Dunn Index [4] and the Intra-Cluster Variance (ICV) [5] metrics as well as with a new metric, Average Item-Cluster Similarity (AICS), we introduce in this section. The Dunn Index [4] and the ICV [5] metrics evaluate the physical layout of services on the grid. The Dunn index determines the extent to which services belonging to the same cluster are placed closer on the grid than services from different clusters. The ICV metric indicates the level of similarity between the services in a cluster and is computed as the sum of squared deviations between services in a cluster and the cluster center [5].

Table 1 Clustering results for varying number of iterations and Cluster Retrieval Distance

NoIt	CRD	No. of Clusters	Average Cluster Size	Dunn Index	Intra-Cluster Variance	AICS
1 mil	3	280	2	0.013	0.529	0.565
1 mil	4	230	3	0.012	6.626	0.438
1 mil	5	157	4	0.005	11.854	0.294
1 mil	6	31	23	0.018	67.884	0.073
1.5 mil	3	283	2	0.013	0.881	0.533
1.5 mil	4	199	3	0.013	12.08	0.406
1.5 mil	5	145	5	0.018	13.466	0.269
1.5 mil	6	35	21	0.014	51.519	0.098
2 mil	3	276	2	0.011	1.185	0.546
2 mil	4	220	3	0.013	7.302	0.446
2 mil	5	144	5	0.020	7.420	0.264
2 mil	6	58	12	0.005	24.753	0.120

We introduce the AICS metric (formula 18) to verify that the services in a cluster are semantically similar to each other. The metric computes an average similarity value between each service and the other services in the cluster.

$$AICS = \frac{\sum_{s \in S} AVG\_DOM(s,c)}{|S|} \tag{18}$$

where S denotes the set of services available for clustering, C denotes the set of all clusters, $s \in c$ and $c \in C$. The function AVG_DOM returns the average DoM between a service and the other services found in its cluster and is defined as follows:

$$AVG\_DoM(s,c) = \frac{\sum_{s' \in c, s' \neq s} DoM_{services}(s,s')}{|c|-1} \tag{19}$$

The theoretical minimum number of iterations for a number of 737 services we used from the SAWSDL-TC benchmark is 1.5 million. However, the results in Table 1 show that reasonable results can also be obtained by running 1 million iterations of the clustering algorithm. The cluster retrieval distance (CRD) is used to determine whether a service should be considered as part of a cluster judging by how near it is placed to that cluster. Results show that, for the SAWSDL-TC benchmark, a cluster retrieval distance below 4 leads to precise but very small clusters, while a value above 6 leads to large clusters containing services that are not necesarily similar. The useful range of values for this parameter is thus between 4 and 5 grid units, depending on the desired precision and size of clusters. The Dunn Index should be maximized for good clustering results. This is indeed the case for 1.5 and 2 million iterations, with a cluster retrieval distance of 4 and 5. These are also the cases which provide the most coherent clusters in terms of their content (according to AICS). The ICV should be minimized to obtain compact and well defined clusters. For a cluster retrieval distance of 3, the variance is indeed very small but the clusters mostly contain only two services. For a cluster retrieval distance of 6, the variance is very high and the clusters contain many services that are not similar to each other. Again, values of 4 and 5 for the cluster retrieval distance seem to provide a good compromise. Overall, the results can be interpreted as follows: (1) a smaller iteration number associated with a larger cluster retrieval distance leads to larger but less well defined clusters, (2) a larger iteration number associated with a smaller cluster retrieval distance leads to smaller and better defined clusters.

6 Conclusions

In this paper we have proposed a method for clustering semantic Web services inspired by the behavior of ants. The method clusters services based on their semantic description. A semantic service description includes ontology concepts annotating the service input and output parameters. We propose a method as well as a set of metrics to evaluate the degree of semantic similarity between service descriptions. We have tested the ant-inspired clustering method on the SAWSDL-TC benchmark. The experimental results evaluated with the Dunn Index, the Intra-Cluster Variance and AICS, an original metric we introduce in this paper, validate the clustering method and demonstrate its efficiency.

References

1. Abraham, A., Das, S., Roy, S.: Swarm Intelligence Algorithms for Data Clustering. In: Soft Computing for Knowledge Discovery and Data Mining, pp. 279–313. Springer, US (2007)
2. Charles, et al.: On the implementation of SwarmLinda. In: Proc. of the 42nd annual Southeast Regional Conf., pp. 297–298. ACM, New York (2004)
3. Deneubourg, J.T., et al.: The dynamics of collective sorting: Robot-like ants and ant-like robots. In: Proc. of the Inter. Conf. on Simulation of Adaptive Behavior: From Animals to Animats, pp. 356–363 (1991)
4. Halkidi, M., Vazirgiannis, M., Batistakis, Y.: Quality scheme assessment in the clustering process. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 265–267. Springer, Heidelberg (2000)
5. Handl, J., Knowles, J., Dorigo, M.: Ant-Based Clustering and Topographic Mapping. Artificial Life 12(1), 35–61 (2006)
6. Liu, W., Wong, W.: Web service clustering using text mining techniques. International Journal of Agent-Oriented Software Engineering 1(3) (2009)
7. Marinakis, Y., et al.: A Hybrid Clustering Algorithm Based on Honey Bees Mating Optimization and Greedy Randomized Adaptive Search Procedure. In: Maniezzo, V., Battiti, R., Watson, J.-P. (eds.) LION 2007 II. LNCS, vol. 5313, pp. 138–152. Springer, Heidelberg (2008)
8. Nayak, R., Lee, B.: Web Service Discovery with additional Semantics and Clustering. In: Proc. of the International Conf. on Web Intelligence, pp. 555–558 (2007)
9. Platzer, C., et al.: Web Service Clustering Using Multidimensional Angles as Proximity Measures. ACM Transactions on Internet Technology 9(3) (2009)

10. Skoutas, D., et al.: Efficient Semantic Web Service Discovery in Centralized and P2P Environments. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 583–598. Springer, Heidelberg (2008)
11. Wong, W., Liu, W., Bennamoun, M.: Tree-Traversing Ant Algorithm for term clustering based on featureless similarities. Data Mining and Knowledge Discovery 15(3), 349–381 (2007)
12. SAWSDL-TC, http://projects.semwebcentral.org/projects/sawsdl-tc/

A Distributed Service-Based Software Infrastructure for Trip Planning in Motorway Auto-routes Context

Alberto Grosso, Davide Anghinolfi, Massimo Paolucci, and Antonio Boccalatte

Abstract. This paper presents TRIPLAN, a distributed service-based software architecture providing the European travellers, who are planning their journeys, with high quality information in order to make them taking decisions about trip planning, by taking into account both predicted traffic conditions and real time ones. The TRIPLAN architecture is solution based on the following criteria: use state-of-the-art and innovative methodologies and techniques; exploiting the available information for providing more effective advices; enforcing standards for information exchange and cooperation among different systems in TERN; assessment and continuous upgrade of a knowledge base on traffic conditions; provide a set of architectural guidelines for a service information network. The focus of the work presented in this paper is the design of the SW architecture for providing via Web a traffic planning service based on an integrated informative system.

1 Introduction

The aim of this paper is to define a software architecture able to provide services for travel planning (Trip Planning Services -TRIPLAN) promoted through the Web. The architecture should be able to integrate heterogeneous information sources that are geographically distributed and whose availability may change over time. Based on these information sources, the system should be able to supply different planning services primarily divided into pre-trip and on-trip planning to be maintained over time even in update feature of the algorithms that underlie those services. Considering the distributed nature of the system, the heterogeneity of the information to be integrated, and the decoupling between the services offered and client applications, it seems reasonable to adopt the design guidelines provided by the service-oriented

Alberto Grosso · Davide Anghinolfi · Massimo Paolucci · Antonio Boccalatte
Department of Communication,
Computer and System Sciences University of Genova,
Via Opera Pia 13, 16145 Genova, Italy
e-mail: {agrosso,nino,paolucci,anghinolfi}@dist.unige.it

architecture software engineering. Moreover, since the standardization work is still ongoing, with the aims of defining a platform-independent model for exchanging traffic information called Datex II within the Easy Way [1] project, the system architecture of TRIPLAN must be compliant with this specification and integrate the mechanisms and protocols for exchanging data therein. Taking into account the above considerations, the paper is organized as follows. After a brief introduction, the compliancy with Datex specification is briefly discussed. Then, the service-based architecture along with the integration of distributed data sources and external components are detailed. Finally, conclusions and future activities are provided.

2 The Distributed Software Architecture for Supporting Trip Planning System

2.1 DATEX Specifications

According to the technologies employed in nowdays applications for providing traffic information services, different formats and mechanisms for the transfer and representation of traffic data have been considered [2]. A leading role in this field is represented by the DATEX specifications, these specifications are defined by the support of the European Commission with the aim of exchanging information among traffic management centers, centers for traffic information, and service providers. Datex aims to standardize the interface for communication between control centers and information centers of traffic. The second generation of the DATEX II specification started in December 2006 from the corresponding website (www.datex2.eu), and now it is promoting the involvement in the project of all the actors working in the field of traffic and travel. With the new DATEX II generation the attention is directed to all the applications that require dynamic access to information related to traffic and travel in Europe [3]. DATEX II promotes itself as multi-part technical specification and it is maintained by the CEN Technical Committee 278 (relating to Road Transport and Traffic Telematics). More information about the latest version of these specifications are available from the DATEX II website. The standard implies a specific model for exchanging information through Web Services via HTTP (Exchange PSM) based on a model platform for the exchange of data (Exchange PIM).

2.2 Detection and Classification of the TRIPLAN Services

The services within the TRIPLAN architecture are arranged in layers depending on the type of logic that they encapsulate, the potential reuse of their underlined logic, and the reference to the elements that shape the application domain. Therefore the services are classified in: Entity Service, Task Service, Utility Service. The first group represents the entities relevant in the context of highway travel planning; this category covers: services related to the data sources of the traffic information (e. g. the service providing the amount of trucks passed through a given path in a given period), services related to find alternative paths or the shortest paths on a given route, services referred to the weather conditions, and those providing information about events and fairs taking place in the territory of interest. The Entity Services can be modeled either as consistent simple services or as a composition of different basic services. Considering the Task Service category, it involves application services that implement the business logic of the system, hence those that can be consumed directly by the client applications of the TRIPLAN system. Therefore, this category includes all the planning services, short and long term, and all the functions related to them (user preference settings, alerts and notifications management, etc.). Finally, the infrastructure services of the system, called Utility Services, include services for the management of security aspects (encryption, certificate management, etc.), authentication services, transaction management, logging and exception handling.

2.3 Composition and Service Control

When designing the services there is the need to choose the standards to be included in the SOA and and the way for implementing them to better support the features and requirements of the service-oriented solution [4]. Considering the definition of contracts and protocols for data communication at the Entity Level, we decide to refer to the Datex specifications [5]. The role of service composition is the management and evolution of so-called composite application that embody the business logic of the system processes [6]. The criteria for the services composition may be multiple, as a basic composition strategy for the TRIPLAN system you can consider an aggregation over time and space, this choice promotes the composition of services within the same category (for example, consider services that map the traffic data sources) which relate to the same section of a motorway (aggregation in space) or that elaborate data reffered to the same time interval (aggregation in time). Service composition remains transparent with strategies that can be extended. For supporting the composition phase, a dedicated Control Service is integrated with the process, its main purpose is to implement, manage, and monitor the services offered in the SOA. The Control Service includes the monitoring activities for displaying the flows of business events issued by corporate business processes, including: Business Activity Monitoring, Identity and Access Management, Surveillance Service, Service Level Management, Service Registry Repository, and Execution policy.

2.4 Service Oriented Business Processes

After establishing an inventory of basic services, we proceed with the creation of the orchestration layer. The orchestration includes the management of transitions among individual services, including errors handling, as well as the description of the overall process [7]. Hence, SOA business processes are formed by flows of basic services and composed services related to:

- provide information about traffic: historical, predictive, or real time;
- provide information about the presence of events in the territory;
- provide meteorological information;
- optimization algorithms;
- implicit or explicit utility services execution.

In order to define and execute these processes an orchestration engine (workflow manager) is integrated in the architecture. In this context, we select the features to be supported by the architecture; these features are reduced to the corresponding utility services. This helps us to decide which aspects of the WS-* specifications will be part of service oriented environment. The goal is to create a layer on the ubiquitous XML messages whose architecture is built on two principles: a message-oriented approach, the need to deal with technlologiacally independent communications. The specifications selected as infrastructure services of the system (Utility Services) are: WS-Addresing, WS-Policy, WS-Security, and WS-AtomicTransaction.

2.5 The Service Oriented Architecture of the TRIPLAN System

This section details how the elements described in previous sections are combined together to form the SOA TRIPLAN system. The services classified as Entity Service, namely those related to the basic entities in the application context (traffic data sources, motorways routes management, weather forecasts, etc.), are organized in the Data Service Inventory and interoperate each other through a dedicated bus. Considering the nature of those services and the need of the planning system to historicize the information referred to specific situations of interest, such as the presence of a significant flow of traffic with a given weather conditions, the data provided by these services will be structured and persisted on a dedicated repository: the Traffic Data Repository. In the same way, the infrastructure services (e. g. authentication, transactions management, etc.) are arranged in another repository called Utility Service Inventory. Thanks to the composition service, it is possible to aggregate the services made available by each respective inventory in order to provide complex functionality [8] which are more relevant to the context of travel planning, resulting in what we call Traffic Service. In this context, the features related to heterogeneous data sources, but connected to the same motorway section, will be aggregated with specific business logic and offered as a single service. The integration obviously may involve different utility services and historical data, for example for providing the value of an indicator about the traffic flow on a given route. Finally, the SOA of the TRIPLAN system provides client applications with the trip planning services. These services are modeled as workflows of basic and composed services and they integrate the functionalities of the optimization algorithms that cover one of the focal point in the TRIPLAN system. The TRIPLAN architecture includes the presence of an Enterprise Service Bus [9] through which all services available in the Data Inventory Service can be consumed, composed, or involved into complex business processes with the support of security policies, authentication, transportation, etc. Obviously, the infrastructure services available in the Utility Service Inventory will be integrated on the bus and will operate through explicit invocation or through the application of specific policy [10]. Processes and algorithms for planning and optimization will be managed within a dedicated architectural level and modeled as workflows. A dedicated engine operates the execution of the workflows by invoking the services through the Enterprise Service Bus. In order to integrate in the TRIPLAN system existing web services or legacy applications the software designer can just use (if available) or simply implement (if not yet implemented) a dedicated adapter; the purpose of the adapter is to promote in the SOA the functionalities of the system to be integrated by ensuring the support of the data communication protocols compliant with the adopted standards.

3 The Presentation Layer

In this section the design of the software presentation layer compliant to the service-based architecture defined for the TRIPLAN system is detailed. Following the considerations reported above, the proposed TRIPLAN Web Interface software infrastructure is composed by a dedicated Presentation Business Logic tier (PBL tier) and by a set of user interaction services (UI services). The role of the Presentation Business Logic access tier is to provide access to the Service-Oriented business logic by abstracting the invocation of the Business Logic services; it can also manipulate the data results offering a programmatic interface that could be consumed by the UI services. The PBL is instantiated a single time in the server side and communicate with the client side through AJAX technology. The User Interaction services model the interoperation with the user; each service is bounded with a dedicated presentation interface and one or more specific services of the PBL. The UI services can be composed within a web portal and so consumed through the web by the users. In order to manage user profiling issues, n-instances of the UI services could be instantiated for user group or membership.

4 Conclusions and Future Activities

It can be stated that the implementation of the architecture proposed for TRIPLAN system is a complex process, but considering the high diffusion of SOA in the development of enterprise applications, the times and costs for implementing the system can be reduced through the adoption of development instruments and dedicated middleware components [11] that make some development phases automatic and provide many of the infrastructure services designed in the proposed architecture. The travel planning services offered by the TRIPLAN software system are related to highway routes on which the system focuses the analysis of traffic information and the trip optimization algorithms. This scenario however does not restrict the use of the proposed service-based architecture to the motorway context; infact the services offered by the system can be easily extended through the integration of new components, such as new data sources, novel workflow definition, or more general new services. Considering the low coupling of the TRIPLAN software components, the ability to organize and structure complex workflows in basic services (reusability), and the independence versus the client applications, the TRIPLAN system could be extended to work in a multi-modal transport context at low cost.

References

1. EasyWay Project: An integrated approach to major road network (TERN) management. Project for Europe-wide ITS deployment on main TERN corridors, http://www.easyway-its.eu
2. Datex II: The standard for the exchange of traffic related data, latest release on (July 1, 2009), http://www.datex2.eu
3. Mechin, J.P.: PUSH-PULL proposal for Datex2. In: 1st European EasyWay Conference, Giardini Naxos, Taormina (2008)
4. Kaltwasser, J.: DATEX II Improved availability of real-time information services on the TERN. In: 1st European EasyWay Conference, Giardini Naxos, Taormina (2008)
5. Curbera, F., Leymann, F., Storey, T., Ferguson, D., Weerawarana, S.: Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More. Prentice Hall PTR, Englewood Cliffs (2005)
6. Erl, T.: SOA Principles of Service Design. Prentice Hall PTR, Englewood Cliffs (2007)
7. Erl, T.: Service Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Englewood Cliffs (2005)
8. Lankhorst, M.M., et al.: Enterprise Architecture at work, Modelling, Communication and Analysis. Springer, Heidelberg (2005)
9. CollabNet: OpenESB, The Open Source ESB for SOA and Integration, https://open-esb.dev.java.net
10. Ueno, K., Tatsubori, M.: Early Capacity Testing of an Enterprise Service Bus. In: Proceedings of the IEEE International Conference on Web Services, pp. 709–716. IEEE Computer Society, Washington (2006)
11. Bahree, A., Cicoria, S., Mulder, D., Pathak, N., Peiris, C.: Pro WCF: Practical Microsoft SOA Implementation. APress (2007)

WebKM - Online Data Mining System

Mihai Gabroveanu, Mirel Cosulschi, and Nicolae Constantinescu

Abstract. The problem of using legacy information systems by making their services publicly available from the Web has become a tedious one due to the diversity of platforms (operating systems, programming languages, database management systems) on which they were implemented. In this paper we present an online system, *WebKM*, following the client–server paradigm, that allows executing data mining tasks from a thin client (Web Browser). An user is thus released from the management care of local installed software.

1 Introduction

The problem of using legacy information systems by making their services publicly available from the Web has become a tedious one due to the diversity of platforms (operating systems, programming languages, database management systems) on which they were implemented.

Several solutions have been proposed to the problems induced by the usage of legacy information systems. These solutions can be grouped into three categories [11]: *redevelopment* – supposes rewriting the current modules, *wrapping* – wraps an existing component in a new, more accessible software component, *migration* – moves the legacy information system to a new environment, more flexible, preserving the original systems's data and functionality.

The HTTP (Hypertext Transfer Protocol) has gained a tremendous success, due to its simple request/response protocol, becoming the perfect application-level protocol for distributed and collaborative systems. The major drawback of HTTP is its stateless property, overcame during the past years by some technical solutions (usage of HTTP cookies, session variables, URL rewriting, etc).

Nowadays, when useful information is considered to be one of the main assets, the Web has developed and enabled one of the easiest and fastest modality of

Mihai Gabroveanu · Mirel Cosulschi · Nicolae Constantinescu
Department of Computer Science, University of Craiova,
A.I. Cuza, 13, Craiova, Romania
e-mail: {mihaiug,mirelc,nikyc}@central.ucv.ro

accessing it i.e. Web services that employ various middle tier business logic and user-friendly query interfaces.

Taking into account the major revilement of the business area on the literature and its tendency to permeate different communities and domains, it is natural to discuss its conceptual aspects that seems to fit our problematic. Therefore, one goal of our paper is to introduce and to consider our work also in the context of Service-like Oriented Architecture (SOA) [9] paradigms.

Trends in knowledge discovery technology challenge the assumption that works still have to be done i.e. customizing or improving classical algorithms and developing new ones in the area of data mining. We propose a technical solution for allowing users to access properly formed data from various locations (the resources shall be identified by their URLs) and apply different data mining algorithms, all in an easy and transparent manner.

In this paper we present an online system, *WebKM*, following the client–server paradigm, that allows execution of data mining tasks from a thin client (Web Browser).

The rest of this paper is organized as follows: Section 2 presents previous works with references to our approach. In section 3 it is described the application functionality while the last section discusses conclusions and future works.

2 Previous Work

Data mining, also known as knowledge discovery in databases, is the process of discovering potentially useful, hidden knowledge or relations among data from large databases. The process of knowledge discovery is a complex one and can be decomposed into an iterative sequence of operations [2]: *data cleaning*, *data integration*, *data selection*, *data transformation*, *data mining*, *pattern evaluation*, *knowledge presentation*.

Weka[1] [3], a toolkit for machine learning and data mining, is an open-source software written in Java that includes a collection of data mining algorithms for classification, regression, association rules and clustering.

One of the time consuming operation in Weka is the classifier task whose goal aims at selecting the proper class for each input data instance. Due to the high computational effort involved by the classification operation it appeared and were imposed the idea of distributing the calculation on multiple machines or processors (parallel and distributed computing) or its execution on a computer providing more hardware resources (client–server approach).

With respect to the previous ideas, Weka4WS [12] is a framework that extends the Weka toolkit for supporting data mining on distributed Grid environments.

In Weka the entire data mining process is performed on a single machine, while the algorithms for classification, clustering, association rules, etc. can be executed only locally. Those algorithms are launched through Weka Explorer, a common GUI (Graphical User Interface).

[1] *Weka (Waikato Environment for Knowledge Analysis)*, `http://www.cs.waikato.ac.nz/ml/weka/`

Weka4WS supports the remote execution of various algorithms implementations like classification, clustering, or association rules, while the data pre-processing and visualization tasks remains locally executed. Thus each data mining algorithm present in Weka toolkit is exposed as a Web Service.

Another framework, GridWeka [13] allows the usage of multiple computational resources to perform distributed data mining operations. The disadvantage of the working environment for GridWeka is that it is an ad–hoc one, despite the Weka4WS environment which is WSRF-compliant.

FAEHIM (Federated Analysis Environment for Heterogeneous Intelligent Mining) [14] approach is more similar to the one implemented into Weka4WS. The majority of the services exposed by FAEHIM corresponds to Weka algorithms' implementations. WekaG [15], an extension of Weka for grid environment, implements the old Grid service technology using Globus toolkit [16].

One of the latest newcomer, DataMiningGrid [17], is an open-source application that provides tools and services for enabling data mining application execution on the Grid. Globus Toolkit 4 implements the main functionalities and services of the middleware layer from DataMiningGrid architecture, being concerned with the core grid functionality.

Finally, it is worthy to mention that the majority of these frameworks are the outcome of open source projects, in which we can easily integrate various particular data mining algorithms. For example, the distributed algorithms for mining association rules presented in [6, 7, 18] can be implemented in GridWeka or DataMiningGrid.

3 Application Overview

WebKM performs all its data mining tasks using a web browser as a client. The main operations implemented into the system, as can be observed from the Use Case Diagram of Data Mining Application are (see figure 1):

1. *Load Data Set* - This operation allows loading on the server the data set to be mined. The data set can be provided as an URL file or can be uploaded via browser from local user machine. The supported format for data set is ARFF (Attribute Relation File Format)[2].
2. *View Data Set* - Following the data fetching step, the user can preview the loaded instance. Viewing the input data at any moment of the process helps to the understanding of knowledge discovery.
3. *Execute Data Mining Task* - This operation includes:

   a. *Choose Data Mining Algorithm* – the user can select an algorithm from the comprehensive set of algorithms available in Weka. Our application supports all Weka implemented algorithms and is prepared for new add-ons: e.g. Classifiers Algorithms (e.g. Naive Bayes [10], C4.5 [8]), Clustering Algorithms (e.g. K-means [5]), Association Rules (e.g. Apriori [1]).

[2] `http://weka.wiki.sourceforge.net/ARFF`

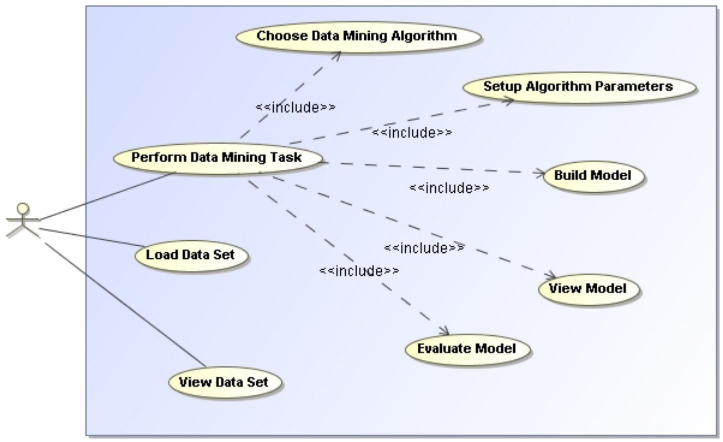


Fig. 1 WebKM - Online Data Mining System - Use Case

b. *Setup the algorithm parameters* – depending on the algorithm selected at the previous step, it is provided an interactive form that allows parameters' tuning.
c. *Build Model* – At this step, the module implementing the selected algorithm is executed and, depending on the previously selected data set, a model is constructed (decision tree, association rules, etc.).
d. *Visualize Model* – the result of running the chosen algorithm for the selected data set with the current parameters set can be visualized in the output format computed by Weka.
e. *Evaluate Model* – the accuracy of computed model is tested by applying it to a new data set having the same format as the build data set.

3.1 Architecture

The system's design can be classified as belonging to a classical *N*-tier web information system [4]. The *N*-tier architectures are the result of connecting many 3-tier systems, adding a layer specifically to allow clients accessing the system through a Web server. In the beginning, the layer for the Web access was placed outside the system, while today, it started to be incorporated into the presentation layer that resides on the server side. The addition of the Web layer led to the concept of application servers used to refer middleware platforms that support access via the Web. The Web imposes itself as an access channel and a Web server must be built in the presentation layer.

In Figure 2 is presented the system architecture design composed from the following components:

1. *First layer*: *Client Browser* – In these systems, the client is a browser and the presentation layer is distributed between the Web browser, Web server and the software that uses data from the application logic to prepare the HTML pages.

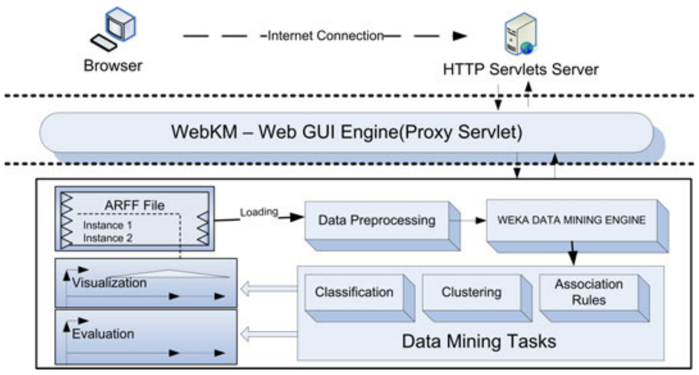


Fig. 2 WebKM - Online Data Mining System Architecture

2. *Second Layer*: *Proxy Servlet* – Since it is fairly complex to develop a presentation layer that includes the Web server, this is considered a true added tier. The application logic layer resides on the intermediate tier. This provides an infrastructure to support the logic of more services integration, and it is called *middleware*.
3. *Third layer*: *Mining Engine* – The lower layer is composed from the service providers that the application logic is trying to integrate. In this case we use *Weka Data Mining Engine*.

WebKM - *Online Data Mining System* was developed using the Java Servlet Technology[3]. Introducing a new data mining's method consist of implementing/adding a new class to the Weka system containing the algorithm, and also adding the name of the method to a configuration file.

4 Conclusion

In this paper we have described the WebKM - Online Data Mining System that allows to execute data mining tasks from a thin client. The strength of the presented method relies on the fact that it is not necessary to install custom software at the client, that for the input data can be used any data sets following the imposed format and is accessible from the Internet and that any available algorithm from Weka can be employed for the user's need. We have discussed, by means of examples, the performances and result of the implemented system. The experiments performed during and after the development phase of application showed us the approach is feasible and we shall focus on the integration of this service in business process.

Acknowledgements. The work reported was partly funded by the Romanian National Council of Academic Research (CNCSIS) through the grants CNCSIS 55/2008 and CNCSIS 375/2008.

[3] `http://java.sun.com/products/servlet/`

References

1. Agrawal, R., Srikant, R.: Fast Algorithms for Mining Association Rules. In: Proc. of the International Conference of Very Large Databases (VLDB), Chile (1994)
2. Han, J., Kamber, M.: Data Mining: Concepts and Techniques, 2nd edn. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, San Francisco (2006)
3. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
4. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services – Concepts, Architectures and Applications. Springer, Heidelberg (2004)
5. MacQueen, B.: Some Methods for classification and Analysis of Multivariate Observations. In: Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability (1967)
6. Gabroveanu, M., Iancu, I., Cosulschi, M., Constantinescu, N.: Towards Using Grid Services for Mining Fuzzy Association Rules. In: Proc. of the 1st East European Workshop on Rule-Based Applications - RuleApps 2007, IEEE Computer Society Press, Los Alamitos (2007)
7. Gabroveanu, M., Cosulschi, M., Constantinescu, N.: A new approach to mining fuzzy association rules from distributed databases. In: Annals of the University of Bucharest. Mathematics and Computer Science Series, vol. LIV (2005)
8. Quinlan, J.R.: C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
9. Bell, M.: Service-Oriented Modeling (SOA): Service Analysis, Design, and Architecture. John Wiley & Sons, Chichester (2008)
10. Mitchell, T.: Machine Learning. McGraw Hill, New York (1997)
11. Bisbal, J., Lawless, D., Wu, B., Grimson, J.: Legacy Information Systems: Issues and Directions. IEEE Software 16 (1999)
12. Talia, D., Trunfio, P., Verta, O.: Weka4WS: a WSRF-enabled Weka Toolkit for Distributed Data Mining on Grids. In: Jorge, A.M., Torgo, L., Brazdil, P.B., Camacho, R., Gama, J. (eds.) PKDD 2005. LNCS (LNAI), vol. 3721, pp. 309–320. Springer, Heidelberg (2005)
13. Khoussainov, R., Zuo, X.: Grid-enabled Weka: A Toolkit for Machine Learning on the Grid. In: ERCIM News, vol. 59 (2004)
14. Shaikh, A., Rana, O.F., Taylor, I.J.: Web Services Composition for Distributed Data Mining. In: Workshop on Web and Grid Services for Scientific Data Analysis, Norway (2005)
15. Perez, M.S., Sanchez, A., Herrero, P., Robles, V., Pena, J.M.: Design and implementation of a data mining grid-aware architecture. Future Generation Computer Systems Journal 23(1), 42–47 (2007)
16. Foster, I., Kesselman, C.: Globus: A metacomputing infrastructure toolkit. The Int. Journal of Supercomputer Applications and High Performance Computing 11(2), 115–128 (1997)
17. Stankovski, V., Swain, M., Kravtsov, V., Niessen, T., Wegener, D., Kindermann, J., Dubitzky, W.: Grid-enabling data mining applications with DataMiningGrid: An architectural perspective. Future Generation Computer Systems Journal 24(4), 259–279 (2008)
18. Stephanides, G., Gabroveanu, M., Cosulschi, M., Constantinescu, N.: A Distributed Algorithm for Mining Fuzzy Association Rules. In: Proc. of the First International Conference on Web Information Systems and Technologies, WEBIST 2005 (2005)

A New Framework for the Extraction of Contour Lines in Scanned Topographic Maps

Tudor Ghircoias and Remus Brad

Abstract. 3D simulations requested in various applications had led to the development of fast and accurate terrain topography measurement techniques. In this paper, we are presenting a novel framework dedicated to the semiautomatic processing of scanned maps, extracting the contour lines vectors and building a digital elevation model on their basis, fulfilled by a number of stages discussed in detail throughout the work. Despite the good results obtained on a large amount of scanned maps, a completely automatic map processing technique is unrealistic and remains an open problem.

1 Introduction

The growing need for Digital Elevation Models (DEMs) and 3D simulations in various fields led to the development of fast and accurate terrain topography measurement techniques. Although photogrammetric interpretation of aerial photography and satellite imagery are the most common methods for extracting altimetry information, there is a less expensive alternative for obtaining the same or similar data: the use of scanned topographic maps.

The most interesting information shown by these maps are the *contour lines (isolines)*, which are imaginary lines that join different points, located at the same elevation. Topographic maps usually show not only the contours but also other thematical layers which overlap: streams and other bodies of water, forest cover or vegetation, roads, towns and other points of interest, toponyms etc. Unfortunately, contour lines are usually the lowest layer in a map after the background layers. For this reason and not only, the extraction of these isolines becomes very difficult, isolines being often fragmented and affected by unwanted elements (noise).

Tudor Ghircoias · Remus Brad
Lucian Blaga University of Sibiu,
B-dul Victoriei 10 550024 Sibiu, Romania
e-mail: {tudor.ghircoias,remus.brad}@ulbsibiu.ro

Thus, to overcome aliasing and false color problems, [7] are employing scanned topographic map converted into CMYK space, where the value of the K channel is used in contour line detection. S. Salvatore and P. Guitton [5] are taking advantage of the HSV color space to obtain a single color extraction. Working directly on gray-level Chen et al. [9] are getting the contour line segments by extraction of all of the linear features from histogram analysis and supervised classification. In [2] lines are removed from the image using a novel algorithm based on quantization of the intensity image followed by contrast limited adaptive histogram equalization.

Many papers treat only the reconstruction problem, considering the contour lines already segmented and separated from the other layers. The use of Thin Plate Spline interpolation techniques are suggested by [4]. D. Xin et al. [8] also use mathematic morphology to filter the binary image. After thinning the contour lines, they extract a set of key points using the c-means algorithm. S. Salvatore and P. Guitton [3] use the global topology of topographic maps to extract and reconstruct the broken contours. The method is inspired from the previous work of Amenta et al. [1].

Most of the reconstruction methods discussed above use only local information and few include information about the global topology of the contour lines. It results in poor reconstructions, with many unsolved gaps or topological errors (ex. intersecting contours), especially in maps with a high curve density.

2 Application for a Semiautomatic Contour Line Extraction from Scanned Topographic Maps

This application aims to be a solution for a semiautomatic procedure for extracting contour lines from raster images of scanned topographic maps, avoiding the time consuming effects of manual vectorization by reducing the human intervention as much as possible. Furthermore, in order to view the result of isoline extraction, the application builds a 3-dimensional model of the map terrain.

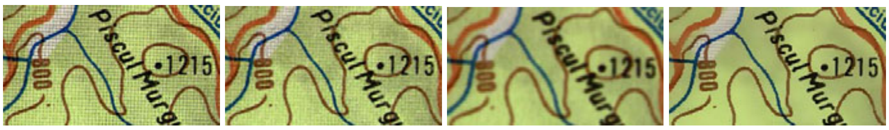
The processing-pipeline of the application is the following:

Scanned topographic map → Preprocessing → Color quantization → Color clustering → *User intervention* → Binary image processing → Skeletonization → Gap filling → *User intervention* → Vectorization → Contour line interpolation → Digital Elevation Model (DEM)

2.1 Color Segmentation of the Scanned Map

Even though the color of the contour lines is not necessary unique, a color segmentation of the scanned map is required for the extraction of all the elements which have the same color with the isolines. But before any segmentation algorithm, a preprocessing of the raster image is indispensable.

The raster image coming from the scanner has usually plenty of noise, originating from the scanning process or from the map itself, which was printed on paper using a halftoning technique. For filtering the scanned image we chose a selective gaussian



(a) Original image (b) Vector median filter (c) Gaussian filter (d) Proposed method

Fig. 1 Noise filtering of the scanned image. As it can be noticed, the last filter has issued the best outcome, despite using a map strongly affected by halftoning.

filter, because it offered the best results without affecting the fine details, such as the contour lines. The algorithm differs somewhat from the classical gaussian filter, because the convolution is applied only between the mask and the corresponding image pixels which have a similar color with the central pixel.

In many cases, the number of different colors in an image is far greater than necessary. In the case of scanned maps, this feature becomes even more evident, because unlike other images, maps consisted of a reduced set of colors before being saved in raster format. A well-known algorithm, fast and with good results on a large amount of images is Heckberts Median-Cut algorithm. Given a certain number of colors wanted to be obtained, the goal of this algorithm is that each of the output colors should represent the same number of pixels from the input image.

The acquired result from the previous step is very useful, because on the basis of the reduced set of colors we can build a histogram, whose local maxima will approximate the color centers, therefore the cluster centers we want to detect. The color histogram provides the input data in the K-Means algorithm. Instead of choosing as initial cluster centers some random color values, the selected colors will be very close to these centers, such that the algorithm can converge rapidly to an optimal solution. These initial cluster centers will be extracted from the local maxima of the histogram.

As color similarity metric we chose the Euclidean metric in the chrominance plane *(a*, b*)* of the *L*a*b** color space. Through the use of the local maxima from the color histogram as the initial data set, precisely these two disadvantages will be eliminated. Not being a random data set, the algorithms convergence is mostly guaranteed. As described in Fig. 2, the application will extract all the pixels from the image that correspond to the cluster of the selected color/colors.

2.2 *Binary Image Processing and Vectorization*

Apart from the contour lines, the binary image created in the previous stage contains many other ‘foreign objects’. As a result, a preprocessing of the binary image is needed for noise removal that would otherwise significantly alter the vectorization result. The noise removal can be accomplished on the basis of minimum surface criteria that contours have to satisfy. As in [8], we used the modified Zhang-Suen thinning algorithm for contour thinning. The algorithm is easy to implement, fast

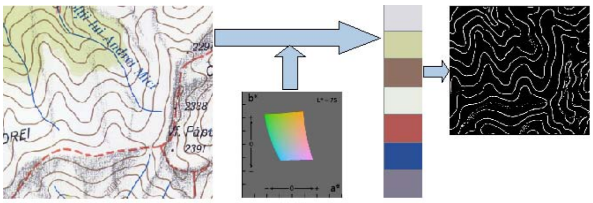
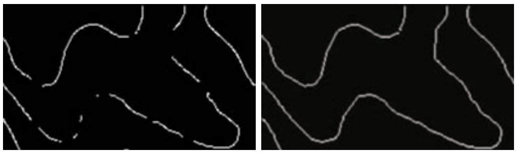


Fig. 2 The user can easily choose one or more colors from the resulting color clusters which best match the color of the contour lines (brown in this case). The resulting binary image will become input in the next stage: contour line vectorization.

and with good results for our problem, where it is critical for the curve end-points to remain unaltered.

One of the most complex problems we came across is the reconnection of the broken contours. Linear geographical features and areas overlap in almost all topographic maps. If two different linear features cross each other or are very close to one another, the color segmentation will determine the occurrence of gaps in the resulting contours. The goal of the reconstruction procedure is to obtain as many complete contours as possible. Every end-point of a curve should be joined with another corresponding end-point or should be directed to the edge of the map.

N. Amenta [1] proposed a reconstruction method which uses the concept of medial axis of a curve λ. Amenta proved that the *crust* of a curve can be extracted from a planar set of points if the points that describe the curve are sampled densely enough. His assumption was that the vertices of the Voronoi diagram approximate the medial axis of the curve.



(a) Original contours (b) Reconstructed contours

Fig. 3 As it can be noticed, not all the gaps are being filled. Therefore, repeating the algorithm for the remaining incomplete contours might be necessary.

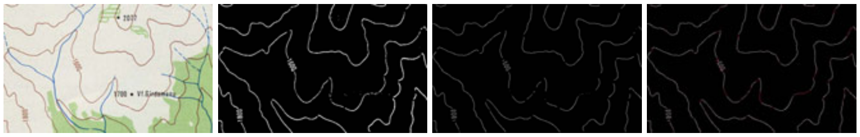
At the end of the reconstruction algorithm the final result of the automatic processing will be presented to the user for a manual correction of the remaining errors and assigning an elevation to each contour line. For the contour line interpolation we followed H. Taud's idea [6]. The isolines can be then saved in a vector format.

3 Results

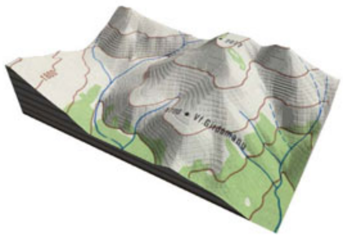
The algorithm was tested on more than 50 scanned images of relatively complex topographic maps, with many colors and overlapping layers. Some of the scanned maps have a poor quality, being affected by noise caused mainly by the printing procedure (halftone) or by the old paper.In the table below, we present an evaluation of the algorithm for 7 different maps. The overall success rate (number of solved errors resulting in complete contours from the total number of errors from the segmentation process) was 83.35%. Out of 64 unsolved gaps or incorrect contour connections 12 came from the elevation values.

Table 1 Algorithm evaluation for 7 different maps

	Quality	Curve Size	No.of contours	No.of gaps	No.of gaps filled correctly	Unsolved gaps	Wrong connec- tions	Success rate
Map 1	medium	thin	9	26	24	2	0	92.31
Map 2	medium	thick	9	26	23	3	0	88.46
Map 3	poor	thick	17	74	57	13	4	77.03
Map 4	high	thin	10	24	18	5	1	75
Map 5	poor	thin	18	38	30	6	2	78.95
Map 6	high	thick	16	50	44	6	0	88
Map 7	poor	thin	52	135	113	20	2	83.7



(a) Scanned map (b) Result of color segmentation (c) Processed and thinned binary image (d) Result of automatic contour line reconstruction



(e) 3D-DEM

Fig. 4 Results

4 Conclusions

We will recall the most important stages, discussed throughout the work:

The quality of the contour line extraction process depends to a great extent on the color processing module. Despite the problems of color image segmentation, using the proposed methods we obtained good results even on maps that were strongly affected by noise. Reducing the complexity through a quantization of the color space and through the use of CIELab color space, more perceptual uniform then RGB, a proper color clustering has been attained. Nevertheless, if the original map is poor, the clustering will fail.

The second stage implied a processing of the resulting binary image from the previous step and the implementation of an automatic contour reconstruction method. The morphological filtering and thinning operations applied on the binary image succeeded to remove most of the noise and unnecessary elements. Because isolines are often the lowest foreground layer of a map, many other features were superimposed on these, leading to interruptions and broken contours. The efficiency of the gap-filling algorithm consists in both a global and local approach. Thus, using the geometric properties of the curves, the achieved results were satisfactory, most of the gaps being automatically solved.

References

1. Amenta, N., Bern, M., Eppstein: The Crust and the Beta-Skeleton: Combinatorial Curve Reconstruction. Graphical models and image processing 60(2), 125–135 (1998), doi:10.1006/gmip.1998.0465
2. Pezeshk, A., Tutwiler, R.L.: Contour Line Recognition and Extraction from Scanned Colour Maps Using Dual Quantization of the Intensity Image. In: Proceedings of the 2008 IEEE Southwest Symposium on Image Analysis and interpretation, pp. 173–176 (2008), doi:10.1109/SSIAI.2008.4512313
3. Pouderoux, J., Spinello, S.: Global Contour Lines Reconstruction in Topographic Maps. In: Proceedings of the Ninth international Conference on Document Analysis and Recognition, vol. (02), pp. 779–783 (2007), doi:10.1109/ICDAR.2007.4377021
4. Soycan, A., Soycan, M.: Digital Elevation Model Production from Scanned Topographic Contour Maps Via Thin Plate Spline Interpolation. The Arabian Journal for Science and Engineering Science 34(1A), 121–134 (2009)
5. Spinello, S., Guitton, P.: Contour Lines Recognition from Scanned Topographic Maps. Journal of Winter School of Computer Graphics 12(1-3) (2004)
6. Taud, H., Parrot, J., Alvarez, R.: DEM generation by contour line dilation. Comput. Geosci. 25(7), 775–783 (1999), doi:10.1016/S0098-3004(99)00019-9
7. Wu, R.Q., Cheng, X.R., Yang, C.J.: Extracting contour lines from topographic maps based on cartography and graphics knowledge. Journal of Computer Science & Technology 9(2), 58–64 (2009)
8. Xin, D., Zhou, X., Zhenz, H.: Contour Line Extraction from Paper-based Topographic Maps. Journal of Information and Computing Science 1, 275–283 (2006), doi:10.1.1.94.8371
9. Chen, Y., Wang, R., Qian, J.: Extracting contour lines from common-conditioned topographic maps. IEEE Transactions on Geoscience and Remote Sensing 44(4), 1048–1057 (2006), doi:10.1109/TGRS.2005.861478

Part III
Collaborative Systems

A Collaborative Approach to Construction of Complex Service Oriented Systems

Ate Penders, Gregor Pavlin, and Michiel Kamermans

Abstract. This paper introduces a new collaborative approach to the construction of large scale service oriented systems using a combination of light weight service ontologies, efficient construction procedures and tools. In particular, machine-understandable descriptions of heterogeneous services with well defined syntax and semantics can be created by multiple designers, without complex coordination of collaborative design processes and without any knowledge of formal ontologies.

1 Introduction

This paper focuses on the methods and tools that facilitate the creation of service oriented systems for information processing in an important class of contemporary applications that require reasoning based on (i) large quantities of heterogeneous information and (ii) substantial domain expertise. In such settings, no single expert has all the required domain knowledge, nor the capability to handle the huge amounts of available information. Furthermore, the complexity of required models and inference makes full automation of the reasoning processes difficult [4]. Instead, the processing of information is carried out collaboratively by multiple experts and automated systems, each providing a specific reasoning service based on special expertise and processing capabilities. In particular,we focus on systems of service providers forming organizations that can be characterized as Professional Bureaucracies [8]. In such systems it is possible to achieve ad-hoc, collaborative processing without the need of a centralized authority on domain knowledge or a shared understanding of all the required processing methods and models.

Under these conditions, the key to an effective combination of expertise and reasoning processes is the use of a proper Service Oriented Architecture (SOA). In this paper we assume the use of the Dynamic Process Integration Framework (DPIF) [3].

Ate Penders · Gregor Pavlin · Michiel Kamermans
Thales Nederland B.V. , D-CIS Lab Delft,
e-mail: {ate.penders,gregor.pavlin,michiel.kamermans}@d-cis.nl

A DPIF-based system handles the automated composition of workflows through service discovery and matching, in which experts and artificial agents providing heterogeneous services, are linked up to collaboratively solve problems. The service discovery and matching processes rely on rigorous ontologies [2] describing processing services and the way in which these services relate.

However, the construction of such ontologies remains a major bottleneck in many real world applications [7]. Common approaches to service composition typically use centralized ontologies [1, 5], which often capture a lot of domain knowledge. This results in complex ontologies with significant ontological commitments. This is especially challenging in the targeted domains, where (i) large numbers of service providers produce and consume very heterogeneous information and (ii) the systems are frequently extended with new services throughout their life-cycles. In such settings traditional approaches to centralized construction of domain and service ontologies are not practical and are likely to be intractable. The complexity of the ontology design has been tackled with the help of collaborative methods, such as for example the approach in [7], where multi-player games are used to stimulate users of the semantic web to collaboratively define ontologies over existing Wikipedia entries.

In this paper we propose a new approach to collaborative construction of large service ontologies, which explicitly considers the properties of the DPIF-based systems and the characteristics of Professional Bureaucracies [8]. In principle, the approach minimizes ontological commitments by exploiting the locality of domain knowledge in the DPIF approach. The approach explicitly takes into account the capabilities of service providers, as well as the fact that in a DPIF system the configuration of processing workflows can be achieved via decentralized local knowledge about the relations between services [3]. The approach requires (i) a simple global service description ontology that serves primarily to align the syntax and semantics of the data that is exchanged between services, and (ii) separate local ontologies that describe the knowledge that experts have about possible relationships between services. The global ontology does not store any knowledge about the possible relationships between services. This type of knowledge is captured by local ontologies, each assigned to a service provider (i.e. an expert or an automated system). In the targeted domains, an expert/service designer knows exactly what types of information are needed in order to provide a particular service. Thus, the relations between very heterogeneous services can be described by the service providers themselves, provided they all use the same language to describe these relations. This is facilitated by the OntoWizzard tool that supports successive participation of domain experts leading to gradually growing rigorous service ontologies which are kept understandable to the experts and designers of automated processes by using human readable keywords and free text.

2 Collaborative Processing in Workflows: Principles

In this paper we target Professional bureaucracies [8]. Such organizations consist of service providers (experts or automated processes) which solve problems in a decentralized manner; i.e. a reasoning process implemented by a service provider is not centrally controlled. In such settings, providers of various types of reasoning services can be viewed as processing nodes in workflows which map simple data, such as sensor measurements and observations from humans, to higher level information delivered to decision makers; i.e. each expert implements a particular transformation between different types of information based on appropriate inference processes and domain models (i.e. expertise). Such a workflow consisting of cascaded processes is often present or at least desired in relevant domains, such as complex crisis management processes.

In this paper we assume that the workflows are implemented with the help of the Dynamic Process Integration Framework (DPIF) [3]. Each expert or an automated process is associated with a DPIF assistant agent, which collects all the relevant information and disseminates the conclusions/estimates to the DPIF assistants of other interested service providers. In other words, DPIF assistant agents put service providers into workflows and facilitate information dissemination.

We illustrate a DPIF-based workflow with an example from the environmental management domain. Figure 1 shows a collaborative assessment of the impact of a chemical incident facilitated by DPIF assistant agents. We assume that the factory staff (FS) can estimate the quantities and the type of a burning chemical. This information is supplied to a chemical expert at the incident location (CE1) to estimate the type and quantity of toxic fumes resulting from the burning chemicals. By knowing the location of the fire, the meteorological conditions as well as the quantity and type of the produced fumes, chemical expert (CE2) can (i) estimate

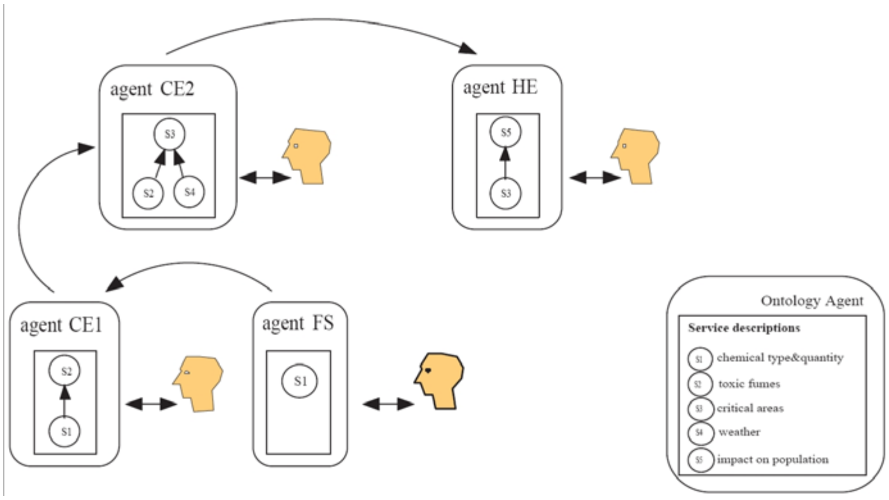
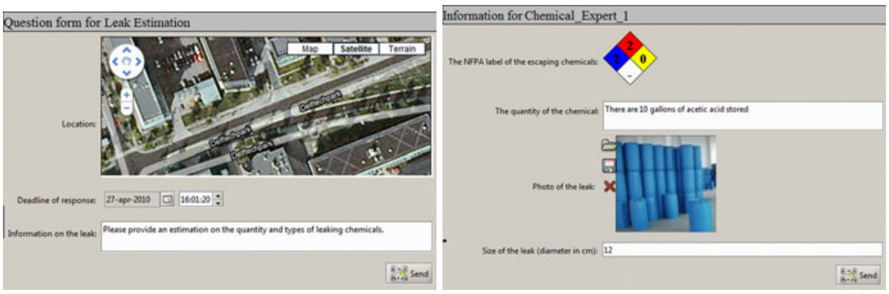


Fig. 1 A DPIF-based workflow in a decision making process. Directed links connecting agents correspond to the information flow between different experts, each processing specific information.

the zones in which the concentration of the toxic gases have exceeded critical levels and (ii) identify areas which are likely to be critical after a certain time. CE2 makes use of the domain knowledge about the physical properties of the gases and their propagation mechanisms. A map showing the critical area is supplied to a health expert (HE) who uses the information on population obtained from the municipality to estimate the impact of the toxic fumes on the humans in case of exposure. Note that, for the sake of clarity, the used example is a significant abstraction of real crisis management processes.

Creation of workflows and routing of information in DPIF is based on the relations between different types of services provided by experts or automated algorithms. An important feature of the DPIF is that these relations are not represented in centralized ontologies. Instead, the relations between different types of services are captured by local ontologies dispersed throughout different DPIF agents. The DPIF approach assumes that each service provider can declare the inputs and outputs of the contributed services, which is sufficient for the automated creation of globally meaningful work flows by using service discovery [3]. In other words, a new expert can participate in a DPIF based organization by mere specification of the provided services (i.e. the outputs) and the inputs required for the delivery of each service. The assistant agents automatically register the specified services in the DPIF system and collaboration workflows are configured at runtime through service discovery. This requires matching of services. Consequently, correct specification of services is critical; all agents supplying or using the same service must use identical service descriptions. In other words, the services have to be described in a standardized way.

Moreover, each type of services is associated with a specific Query form (QF) and a specific Reply form (RF), which streamline the information acquisition and provide rigorous semantics. The Query form (QF) is a specific collection of standard GUI elements which describe the service invocation conditions. The values of these elements *are used as conditions for the discovery of suppliers of the required inputs as well as for the negotiation in which collaboration contracts between services are established*. The Reply Form (RF) is a collection of standard graphical elements and input fields which facilitate the formulation of the service outputs by using well defined semantics and syntax. Figure 2 shows examples of a QF and an RF for the service FS, which supplies information on the leaking chemical, such as the type, the quantities, etc. In the workflow shown in Figure 1, the CE1 uses the QF to request the information on the type and quantities of the leaking chemical. The DPIF uses the data in the QF as conditions in finding a service provider for the requested service. The QF is shown in the GUI of CE1 where he can fill in the values, such as the incident location, time interval and an additional question about the kind of requested information. In this example, the location and time interval are used for the negotiation between the CE's and FS's agents, respectively. After the FS gathers the requested information, she fills in the RF shown in figure 2.


Question form for Leak Estimation

Location:

Deadline of response: 27-apr-2010 16:01:20

Information on the leak: Please provide an estimation on the quantity and types of leaking chemicals.

Information for Chemical_Expert_1

The NFPA label of the escaping chemicals:

The quantity of the chemical: There are 10 gallons of acetic acid stored

Photo of the leak:

Size of the leak (diameter in cm): 12

Fig. 2 a.) Query form for the Leak Estimation service; b.) Reply form the Leak Estimation service

3 Service Description

Typically, services are provided by many stakeholders from different organizations whose capabilities evolve over time. Consequently, large systems of descriptions have to be maintained and it is very difficult to specify a complete set of services in advance. In other words, traditional approaches based on a rigorous centralized ontology capturing all relevant services defined prior to putting the system into operation are not practical; we simply do not know which relevant services will be available in the future and maintenance of large ontologies is likely to be very expensive or even intractable. However, it turns out that in systems characterized as Professional Bureaucracy, we can achieve efficient creation and maintenance of large repositories of service descriptions by adopting the following principles:

- The ontologies describe only capabilities and needs of each expert/node, i.e. provided and required services, respectively. The ontologies do not describe the domain models. We assume that the experts/automated processing nodes possess such knowledge.
- Responsibility for the specification of a particular service is delegated to the providers of that service, i.e. the experts who have local domain knowledge and they know for each of their roles the associated capabilities and the needs.
- The service description consists of three main components: (i) a verbal description and keywords, (ii) a description of the service invocation conditions and (iii) a description of the service outputs. Points (ii) and (iii) are based on a set of objects that represent atomic information types with clearly defined semantics and format.

3.1 Global Service Ontology

All services announced in the system as service descriptions are captured by the service ontology. Each service description is stored as an instance of the 'Service'

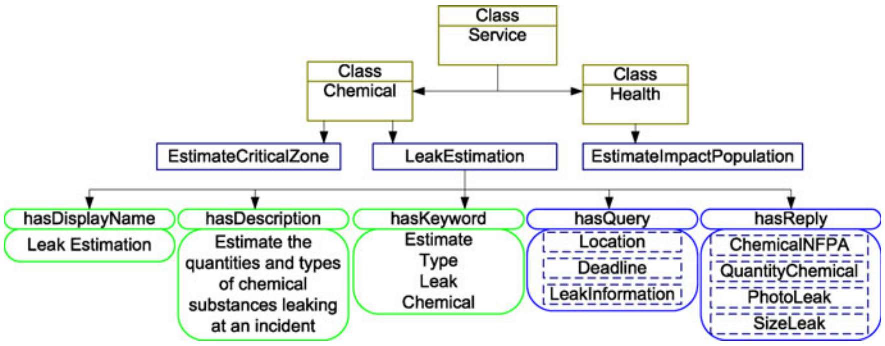


Fig. 3 An example of a service ontology

class and consists of the following attributes: (i) a human understandable name, (ii) a textual description, (iii) a set of keywords related to the service, (iv) a set of elements describing the service invocation conditions and (v) a set of elements describing the service output.

Figure 3 shows an example of a service ontology corresponding to the scenario sketched in section 2. For the sake of brevity we discuss only the 'LeakEstimation' service structure in detail. The 'LeakEstimation' service structure consists of the attributes 'hasDisplayName','hasDescription', and 'hasKeyword', which form a human readable description. Moreover, the clusters of elements called 'hasQuery' and 'hasReply', shown as blue rectangles in figure 3, describe the input and output of the service, respectively. In general, 'hasQuery' and 'hasReply' elements are used for the creation of the QF and RF in the GUIs. The contents of the 'hasQuery' and 'hasReply' attributes define the composition of the forms shown in figure 2.

The global service ontology can be accessed only by a special Ontology agent. As the experts configure and commit their services, the Ontology agent adds these new services to the global service ontology and retrieves information about service descriptions based on keyword and free text matching. Figure 1 shows an example of the Ontology agent, with access to all service descriptions used by experts in the illustration.

Moreover, the elements of the QF and RF are represented as fields that are instances of the'Parameter' class. Each element is defined through the following attributes: (i) a label indicating the type information supported by this field, (ii) a default value, (iii) the type of the field as it will appear in the Query and Reply forms, and (iv) the expected notation format of the field. An example of instances of the 'Parameter' class used in the running example is shown in figure 4. The shown instances reflect the entries in the 'hasQuery' and 'hasReply' attributes of the service instance shown in figure 3. For the sake of brevity only the field 'LeakInformation' is described in detail.

The types of the fields in the QF and RF are defined with the help of the information model, which prescribes a limited set of field types, and their formatting constraints (see example in Figure 5). The information model is a predefined, fixed ontology defining two distinct groups (Classes) of elements:

- The 'FieldType' class, which contains the set of all types of attributes that represent the fields used in QF and RF, such as single line text field, image, map for location specification, list of ADR-labels for chemical substances, etc.
- The 'Notation' class, which predefines the set of possible values for each FieldType. Each element of the Notation class must have a 'Notation Format' to allow the User Interface to verify the entered values.

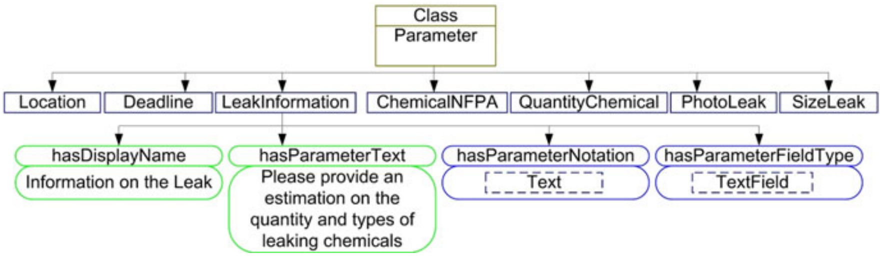


Fig. 4 Parameter instances used in the service ontology shown in figure 3

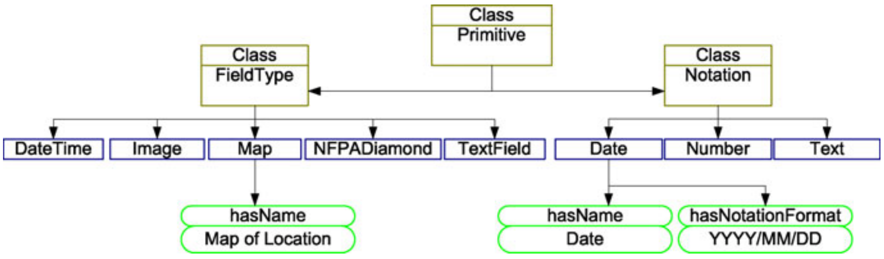


Fig. 5 An example of an information model described in OWL

3.2 *Local Task Ontologies*

The presented approach exploits the fact that workflows in DPIF are constructed by using the knowledge about relations between different services, which is distributed throughout a system of experts/processing modules; i.e. the DPIF establishes a mapping between compatible services by using local knowledge. This is a consequence of the application types, where each expert/process implements a particular transformation between different types of information based on appropriate inference processes and domain models. Thus, we can make a safe assumption that each service provider can specify (i) the types of services she contributes to the system and (ii) the types of services that are required from other experts to provide each of the declared service types.

For each contributed service type, the corresponding DPIF assistant agent stores a local Task Ontology (TO), encoded as an OWL file that captures the relations between the provided service and the required services. The TO contains a copy of the different types of service descriptions used by the expert, to prevent unnecessary requests for information to the Ontology agent, and is only available to the assistant agent of the expert. The boxed directed graphs inside each agent in Figure 1 represent local ontologies associated with the assistant agents in this example. The arrows indicate relations between the different service types: services corresponding to the leaf nodes provide inputs the service provided by the respective agent.

A local TO is an instance of the class 'Task', which has attributes 'hasSubGoal' and 'hasGoal', corresponding to the collection of required services and the provided service, respectively. Figure 6 shows an example of a TO.

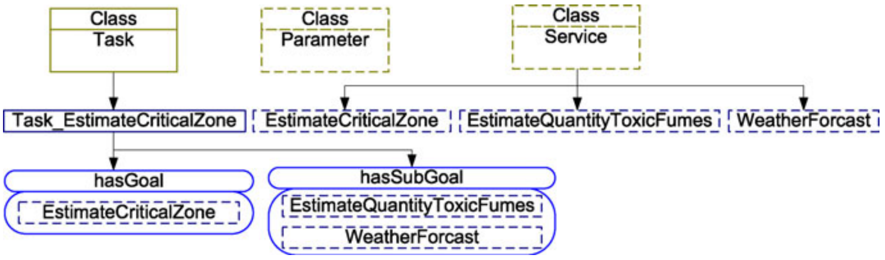


Fig. 6 A fraction of a local task ontology

Each DPIF assistant agent announces their provided and the required services, which are captured by the TO, at a special agent that acts as *yellow pages* with services. This process is the basis for the creation of workflows through service discovery at runtime. Moreover, the TO is used by DPIF agents to create local task objects, the basic building blocks of workflows. As an expert notifies the assistant agent that he or she is going to provide a certain service, the agent finds the local TO that corresponds to the specified service in its local repository.

4 Collaborative Construction of Service Ontologies

Collaborative construction of service ontologies is a key element of the proposed solution to tractable generation and maintenance of large repositories of service descriptions in constantly evolving communities of experts. We assume that in the targeted applications, service ontologies can be built gradually. When a DPIF system is first started, the service ontology is empty, with service descriptions added by the Ontology agent both when new experts with new capabilities join the system, as well as when already joined experts become capable of performing new services. The service ontology is open ended and can thus accommodate new services as they become available throughout the life-cycle of a system.

4.1 Alignment of Service Descriptions

An important aspect to the successful creation of user-generated service ontologies is being able to effect a (qualitative) standard that is understood and adhered to by all users of the system. In the presented approach, where the service ontology is empty when the system is first started, the quality of the service descriptions can only be refined over time as experts join the organization.

These *qualitative* standards must be taken into consideration by an expert during the configuration of local tasks (i.e. at design time). Effective configuration of workflows between experts is possible only if the services bundled in a local task description are described in such a way that they are understood by the community. Namely, a service that one expert describes in keywords and free text must be unambiguously understandable by other experts who either require that service, or supply information that is required by that service.

Making sure that service descriptions remain unambiguous is achieved by exposing experts to potential overlaps or conflicts in their service description when compared to the list of already defined service descriptions found in the Service Ontology. The Service Ontology repository is inaccessible to the users, and is automatically maintained by the Ontology Agent. This agent is also responsible for searching the Service Ontology based on keywords and free text for service descriptions that seem similar to what an expert has indicated their service description is at design time, before experts are allowed to participate in the organization.

Note that the configuration of workflows at runtime does not rely on this central Service Ontology, with which experts only interface during the service configuration. The central Service Ontology is used only at design time, to facilitate consistent service descriptions.

4.2 OntoWizzard: A Configuration Tool

The creation and use of aligned service descriptions is facilitated through the OntoWizzard tool, which supports rigorous specification of services with clear semantics and format, by efficiently exploiting automated search and human capabilities to comprehend natural language. The OntoWizzard consists of a graphical user interface (GUI) that helps the user define a TO.

By using the OntoWizzard's GUI, the expert specifies keywords and describes a service he or she provides using natural language and keywords. These text and keywords are passed on to the assistant DPIF agent, which forwards them to the Ontology agent. Next, the Ontology agent retrieves from its repository those services that best match the keywords that were specified by the user for their service. This list of matching service descriptions, ranked according to the keyword match, is returned to the DPIF assistant where they are shown on the OntoWizzard's GUI. By clicking on an entry in the list, the expert can see a verbal description of the service as well as the associated QF and RF. At this stage the expert can systematically inspect what an existing service provides and how it can be invoked. If an existing service description from the repository matches the capabilities of the newly joined expert, this expert adopts the existing description.

If an expert is contributing a service that has not already been defined by any expert in the organization, this expert makes a new service description using the OntoWizzard. A new service can be made either from scratch, or by extending an existing service. For example, and expert might extend the service for 'the estimation on the impact of chemicals on the human health' to the service for 'the estimation on the impact of chemicals in case of eye contact', and may wish to add a number of extra keywords, edit the human-readable description, as well as add an arbitrary number of information fields for the QF and RF. In case of an extension, the new service should contain at least the same information as the original service, meaning keywords, QF fields and RF fields cannot be removed. In this way, using the OntoWizzard tool in combination with Service Oriented Architectures supports the efficient creation of systems in which services can easily be combined into very complex processing systems, while at the same time maintaining a high standard of quality for service descriptions.

5 Discussion

This paper introduces a collaborative approach to construction of service descriptions in Service Oriented Systems based on the DPIF architecture [3]. With the help of the DPIF, the reasoning about some domain is implemented by self-organized systems of heterogeneous processing services, provided either by human experts or automated systems. In the DPIF approach, the domain knowledge is encoded through local functions/experts. Because of this we can avoid problems typical for the traditional approaches to constructing centralized ontologies that describe domains and complex relations between the services. In real world applications involving many types of services and relations, the maintenance of such centralized ontologies is likely to be intractable. Instead, we introduce an approach where services and relations between these services are described in two separate, *light-weight* ontologies:

- The global service ontology merely captures service descriptions, with a rigid definition of the syntax and semantics of information that is used in service invocation and dissemination of service results. The purpose of this ontology is to align service descriptions provided by different participants at the design time.
- Local task ontologies coarsely describe the relations between services as understood by individual participants in the organization, based on their own domain knowledge. Because these relations reflect only the local knowledge of a participant these task ontologies are expressly not shared. It is these local ontologies that support the runtime creation of workflows based on service discovery.

The global ontology is a central element of the service description procedure. In order to make sure that all agents speak the same language, the global ontology captures three types of elements, namely 1) a verbal description of the service to be provided, 2) conditions under which this service is provided to other participants in the domain, and 3) a collection of elements that can be used in service descriptions. While the vocabulary comprises only a small set of rigidly formalized elements, it is rich enough to allow arbitrarily complex service descriptions to be defined. The global ontology is used during the creation of service definitions, facilitating the matching of service descriptions based on the keywords and free text provided by experts. This results in a ranked list of service descriptions, which in turns allows experts to align their service descriptions with those already in use in their organization. Experts can either reuse a service description that matches their services or extend already existing service descriptions to better match their services. If no predefined service descriptions match their own services, a new service description is made. Moreover, the experts specify relations between services they provide and the services they need. These relations are captured by the local task ontologies which enable decentralized service discovery based on limited, but reliable domain knowledge. *By making each expert responsible for the description of the provided services and relations between required and provided services, local collaboration between experts leads to globally coherent, complex workflows, without any centralized configuration authority or centralized administration.*

The presented principles are implemented by the OntoWizzard, a collection of tools that (i) help experts align their services with the services known to the organization their are part of, based on keywords and natural language, (ii) provide an interface facilitating the inspection of the human readable descriptions and (iii) lets experts define service relations as they perceive them in their domain, in the form of task ontologies. By using these tools, experts contribute to the global service ontology and build local task ontologies without having to use a specific formal language, while the tools make sure that the ontologies are rigorously encoded in OWL. In other words, by using these two types of ontologies, in combination with simple construction procedures, rigorous, machine understandable service descriptions can be created without any knowledge of the formal ontology techniques, and without the problems associated with a centralized ontology approach. In fact, the proposed approach supports creation of service descriptions that comply with internal standards which are only partially defined prior to the operation through a light-weight information model. The standards for composite service descriptions, on the other hand, evolve as more expertise is introduced into the system.

Note that efficient construction of service ontologies in the proposed approach is achieved through explicit consideration of the workflow construction principles based on local ontologies in the DPIF. Also the recently introduced OpenKnowledge approach [6] avoids creation of centralized heavy weight ontologies by explicitly taking into account the application requirements. However, the OpenKnowledge framework also requires specification of *interaction models* shared by the collaborating peers. Such interaction models define workflows for each processing task a priory; the OpenKnowledge approach assumes that collaborating peers understand interaction protocols and the processing sequences. This can introduce additional complexity to the system configuration and the construction of ontologies in which services and processes are specified. Since the approach presented in our paper is targeting systems supporting professional bureaucracies [8], we can avoid such complexity. In the targeted domains it is assumed that experts do not have to share knowledge about their local processes, which results in a further reduction of ontological commitments; the DPIF requires a mere specification of the services and their relations in local ontologies, without any a priory description of collaboration processes.

Acknowledgements. The presented work is partially funded by the European Union under the Information and Communication Technologies (ICT) theme of the 7th Framework Programme for R&D, ref. no: 224318.

References

1. Chiu, D., Agrawal, G.: Enabling ad hoc queries over low-level scientific data sets. In: SSDBM, pp. 218–236 (2009)
2. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? Int. J. Hum.-Comput. Stud. 43(5-6), 907–928 (1995)
3. Pavlin, G., Kamermans, M., Scafes, M.: Dynamic process integration framework: Toward efficient information processing in complex distributed systems. In: IDC, pp. 161–174 (2009)
4. Pavlin, G., Wijngaards, N., Nieuwenhuis, K.: Towards a single information space for environmental management through self-configuration of distributed information processing systems. In: Proceedings of the European conference TOWARDS eENVIRONMENT, Prague (2009)
5. Sheng, Q.Z., Benatallah, B., Dumas, M., Mak, E.O.-Y.: Self-serv: A platform for rapid composition of web services in a peer-to-peer environment. In: VLDB, pp. 1051–1054 (2002)
6. Siebes, R., Dupplaw, D., Kotoulas, S., de Pinninck Bas, A.P., van Harmelen, F., Robertson, D.: The openknowledge system: an interaction-centered approach to knowledge sharing. In: Meersman, R., Tari, Z. (eds.) OTM 2007, Part I. LNCS, vol. 4803, pp. 381–390. Springer, Heidelberg (2007)
7. Siorpaes, K., Hepp, M.: Ontogame: Weaving the semantic web by online games. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 751–766. Springer, Heidelberg (2008)
8. van Aart, C.J., Wielinga, B., Schreiber, G.: Organizational building blocks for design of distributed intelligent system. International Journal of Human-Computer Studies 61(5), 567–599 (2004)

Collaborative Distributed Fault Simulation for Digital Electronic Circuits

Eero Ivask, Sergei Devadze, and Raimund Ubar

Abstract. In this paper we focus on the framework for aggregating adaptively computing resources in different enterprises for computation intensive applications. Concept and implementation of web-based collaborative system was presented to speed up fault simulation and to overcome the problem of memory limits in the case of very large digital circuits. Issues of task partitioning, task allocation, load balancing were handled, credit based priority concept was introduced. Experimental results show feasibility of the solution and gain in performance.

1 Introduction

Today, complexity of integrated circuits is still increasing according to Moore's law, which states that transistor density doubles about every two years. This trend is predicted to continue at least for another decade, posing serious testing problems.

Widely used task in modern digital design process is fault simulation, which is used to estimate the fault coverage, i.e the quality of the product. Additionally, procedure of fault simulation is often required for other test-oriented tasks like fault diagnosis, test pattern generation, built-in-self-test optimization, test compaction, design of reliable systems and others. For certain computation intensive tasks like test generation, the intermediate fault simulation step needs to be carried out many times, hence making simulation speed one of the key issues for overall task performance.

One solution would be to improve the fault analysis algorithms, however the abundance of different methods proposed during the recent years leaves little room for further improvement. Another approach for faster simulation would be to cooperate and combine available computer resources. We can assume that participants

Eero Ivask · Sergei Devadze · Raimund Ubar
Tallinn University of Technology, Raja 15,
12618 Tallinn, Estonia
e-mail: {ieero,serega,raiub}@pld.ttu.ee

computers are not fully loaded all the time. Working could be much more effective when every participant had a chance to run his tasks using collective resources, especially when it is possible to parallelize the task execution. Parallelization technique is application specific, though.

Concept of infrastructure of the current solution was initially inspired from MOSCITO [1, 2] used for example in European VILAB cooperation project. Original system was implemented as client-server Java application. It allowed to invoke single work tools remotely and organize them into predefined automated workflows. Users in different locations cooperated via sharing their software tools in joint workflows. However, parallel execution was not supported. More over, major limitation for widespread Internet based use was TCP/IP socket based communication schema which required opening dedicated communication ports in firewalls not only on server side, but also on the client side for each application and for end users.

More flexible web-based solution for remote tool usage following some key ideas of MOSCITO was proposed in [3]. Socket communication was replaced with HTTP, Java Servlets were used on server side and Applets along with Java applications were used on client side, also database was introduced. In current paper, this concept is revised and improved to support distributed computing and collaboration via effective resource sharing.

There are also several general-purpose frameworks for distributed computing. BOINC [4] for example, built with PHP scripts and used for *seti@home* project, AliCE [5] which uses Java based Jini technology. Both have drawback of relying on remote procedure calls, which is restriction in firewall-protected networks. Another project providing support to high-throughput computing is Condor [11], allowing users to take advantage of idle machines that they would not otherwise have access to. Condor is suitable for executing long running jobs that require no user interaction. Application source code must be recompiled with special libraries. Solution is useful to do task allocation and load balancing, it does not provide task decomposition.

The paper is organized as follows: overall concept of web-based infrastructure is described in section 2. In section 3 priorities concept is explained. Task partitioning and task allocation are handled in sections 4 and 5. Section 6 presents workflow with distributed computing. Experimental results are presented in section 7 and conclusions are given in section 9.

2 Distributed Environment

We have used Web-based solution following a multi-tier concept. There is one coordinating server, several workstations and arbitrary number of users (Figure 1). On workstations simulator tool instances are running. Simulator is provided with network communication abilities, it is wrapped with additional software layer, simulator Agent, is created. Agents must run on the same computer as simulator. Coordinator will serve both clients and agents. Users can access only the Coordinator. Clients and Agents work in 'polling' mode, Coordinator is working in 'answering' mode. At first, Coordinator process and Agents must be started by administrators. Thereafter, client can initiate a task which is passed to the Server where it is stored until a free Agent is asking for a new task. Each user has its own server-side workspace in the database. Large files are saved to file system. When task is complete, Agent passes results to Server where results are assembled and stored again until user will ask for results.

2.1 *Implementation*

Infrastructure of the system is built with Java Applet/Servlet technology. Communication flow between system components can be seen in Figure 1. Tomcat is the Servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. Servlets play important role in order to gain access to the intranet resources on the workstations and to the MySQL database. Simulator tool is implemented in C language. Wrapper agent for simulator is written in Java, which has excellent support for network programming. System components can run on different computing platforms. Simulator instances must run on their native platform. Coordinator servlet resides usually separately from agents, they must not reside in the same local area network. Each Agent can reside on different local area network. Since accessing local hard drive for Java applet is restricted for security reasons, then GUI applet has been signed digitally, with so called self-signed certificate for simplicity. Certificate shows owner specific information. When signed Applet is first time downloaded into use's computer, informative dialog box is displayed. User's responsibility is to trust or distrust the origin and content of the Applet. User can contact Applet owner about authenticity of certificate, when needed.

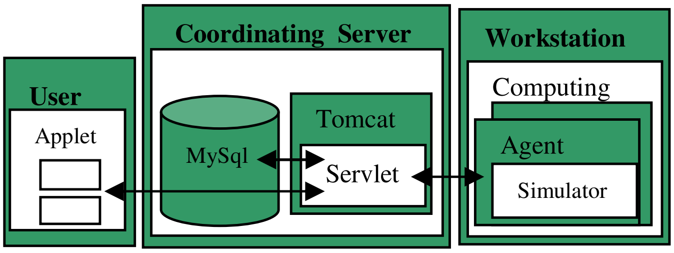


Fig. 1 System components and communication

2.2 *Communication*

Use of applet/servlet approach means that general communication is based on HTTP protocol. Data is exchanged as serialized Java objects (Transfer Object design best practice). Firewall traversal is no problem as only one web server port must be configured on the Coordinator Server. There is no need for opening extra ports in the firewall on the user side as it is the case with other solutions (which would be major restriction). Communication could be secured via SSL encryption, when necessary.

2.3 Data Management

Data handling takes place in coordinator servlet on the Coordinator Server. Problem is that web-based HTTP communication is naturally stateless. Normal HTTP session is valid only for short time, but simulation process may run much longer. We must identify users and store all their data. Users can then come back and receive their results later. Data module has three layers: presentation layer, business logic tier (data base queries, etc.), physical database (MySQL).

First two layers are implemented in Java. User is accessing database via presentation layer, not directly. User tier consists of several functions to run business layer queries. Database access is implemented using Data Access Object (DAO) design practice. Data access is using also connection pooling to speed up DB transactions. Data passing between the user and the Coordinator Server and between the workstations is implemented following Transfer Object (TO) design practice. Relevant information is not sent string by string, but it is passed once as data bundle.

3 Priorities

One important prerequisite for collaboration is that performance of the each participant is enhanced overall, nobody should experience unfair lack of the computing power, especially these participants who previously have donated the most of their resources. This implies we have to introduce a priority system based on each participant credit. *Credit* for host computer is calculated:

$$Credit = Credit + Mode \cdot HostPerf \cdot CompTime \cdot LoadFraction,$$

where *Mode* equals ’1’ when computer is donating and ’-1’ when consuming; *HostPerf* refers to performance index of the computer, it is calculated initially after executing sample calibration task; *CompTime* is the time host computer has devoted to the current task; *LoadFraction* is number between 0 and 1, inclusively. *Credit* rating of the host increases while other participants are consuming its processor time, at the same time ratings of the quests will decrease the same amount. *Credit* numbers can go negative, but it does not interfere, it just shows who is contributing and who is not. Participant with higher credit rating will have higher priority.

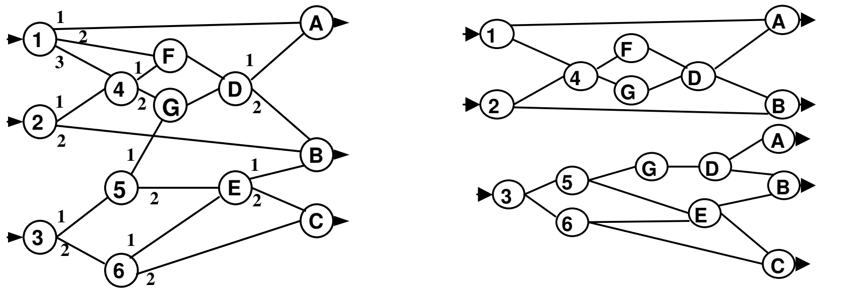
4 Task Partitioning

Generally, there are several methods to parallelize the fault simulation: algorithm can be parallelized, circuit model can be partitioned into separate components and simulated in parallel, partitioning the fault set data and simulating faults in parallel or test pattern data can be partitioned. In this paper, we rely on model parallelism and test set parallelism, faults are included already in simulation model.

Our method of fault simulation based on the parallel exact critical path tracing, together with the detailed method of model partitioning can be found in [6]. The simulation model is created by a special topological analysis of the circuit. Each reconvergent subnetwork of the circuit produces a sequence of computation formulas, and the nested reconvergent subnetworks produce nested computation formulas. The whole set of formulas created in the process of the topological analysis of the circuit represent the whole compiled simulation model. This model allows to carry out fast efficient fault simulation of the circuit where the number of repeated computations is minimized. In [6] it is shown how to divide the process of fault simulation driven by the simulation model into a number of parallel sub-processes. For each sub-process, a partial simulation model is constructed. Creation of such a set of independent simulation models gives us the opportunity to use a distributed environment to achieve higher speed of fault analysis.

As an example, in Fig. 2a a topology of a given circuit is shown where the nodes represent fan-out stems and/or reconverging fan-ins, and the edges represent signal paths of the circuit through other nodes of the gate-network of the given circuit. Such a graph is called a *reconvergency graph* of the given circuit which is the basis for constructing of the simulation model. Reconvergency refers to the case when two signal paths in the circuit have the same origin, and converge in the same node.

Fig. 2b shows a possible solution of dividing the reconvergency graph in Fig. 2a into two overlapped reconvergency graphs which allow to create two simulation models to be processed independently in parallel preserving all the advantages of parallel fault simulation by critical path tracing.



a. Reconvergency graph G of a circuit **b.** Partial reconvergency graphs G_1 and G_2

Fig. 2 Reconvergency graphs

It would be possible to find the optimal selection of partitions to minimize the sizes of overlapped areas. We have used here randomized technique. First we select nodes randomly into partitions, thereafter we count the number of the overlapping nodes. Then we generate a new partitioning and evaluate number of overlapping nodes again. On each iteration, we keep the best solution. The number of iterations is a function of the number of primary inputs of the circuits.

5 Task Allocation

In real-life situations tasks are accumulated and then scheduled in batches [7]. Optimal task allocation ensures that all the computers stop computing at the same time. Several aspects have to be considered: very slow performance machines should get only a small fraction of total load; workstation, which has fault, should get no load at all. Distribution of the tasks should happen within short time interval. Scheduler has to be fast not to delay the execution of the tasks. The scheduling objective is to minimize overall execution time, which is actually time of the longest executing subtask.

There are possibly two conflicting goals while allocating tasks to different computers: 1) maximize the speed of particular task execution for particular user 2) maximize system overall throughput from the perspective of all users. In case of collaborative networking we assume that the latter is more important. Generally, task allocation can depend on several factors that may change in time: number of application users, number of computers, computers workload, speed of the communication links.

Initially, idle agents are polling the Coordinator Server. Each agent announces its allowable load threshold, its current load and current performance index. Agents can determine its performance index by host profiling i.e. executing a small sample task and measuring the completion time. Thereafter, Agents prepare its profiling message for the Coordinator. This message includes also time stamp, so the Coordinator can measure the transfer time and determine network latency index for this particular Agent (it is assumed that clocks of both hosts are synchronized). The Coordinator collects the information initially for some predetermined time, obtained coefficients are applied later- faster machines will get larger tasks respectively.

Let's assume now, that tasks are already submitted to the Coordinator. Scheduler takes the task with highest credit rating and selects an Agent with least load index in the list, given that Agent's load is smaller than allowed threshold. Thereafter, scheduler takes the task with next highest credit rating and searches for next Agent satisfying the threshold condition above. This repetitive process continues until there are no tasks in the list.

The Agent selection idea in this intuitive scheduling strategy corresponds to *Best-fit* algorithm described in [8] and used for memory allocation problem.

When higher-priority task is submitted to scheduler, and there is no idle Agent available in the system, then scheduler has to suspend or reallocate some of the current tasks if expected completion time of the priority task is greater than migration time of the current task. Candidate task for migration is the one who has the lowest priority and longest execution time. Migration means that Agent notifies the Coordinator and sends intermediate result of simulator tool to the Coordinator, thereafter Agent terminates the simulation task. Scheduler finds new target candidate for the task under migration – it will be the Agent with the lowest credit rating and with the best performance rating.

Task must be also migrated when load on the host increases above allowable threshold, i.e. if local computing activity increases, as our goal was that participants in collaboration may not suffer. Exception is only the case, when such task belongs to the host owner. Load threshold is adjustable by each participant. Further information on load balancing can be found for example in [9, 10, 11].

6 Simulation Workflow

First, user specifies parameters and design file location for simulation tool (see Figure 1). In addition, size of the simulation task can be predefined by user. Thereafter user GUI contacts with the Coordinator Server and described parameters along the model are passed automatically. Task coordinator process (Java servlet) records all requests from user(s) to the database. Java based test Agents poll constantly the Coordinator and if any subtask is scheduled by the Coordinator process, then Agents receive the appropriate parameters and model file and will start actual native simulator tool executable.

Simulator first constructs simulation model taking into account of the size limit of the subtask. While reaching the limit, it saves the breakpoint information into local file system. Simulation Agent then reads the breakpoint information and passes it to the Server where it will be stored for other simulation agents. When next idle Agent is polling, then it will receive the circuit file along parameters and breakpoint information. Agent starts new copy of simulator tool on the other computer. Simulator first constructs simulation model again, but this time it is not starting from beginning, but restores from breakpoint up to the point when task size limit will be reached. New break point information will be again saved to local file system and later passed to the Server database by the Agent. Simulator Agents then wait until their subtasks will be completed and report results back to the Coordinator.

Process repeats until there are no simulation sub tasks left. Note that simulators have been started subsequently, but thereafter they run concurrently. Total starting delay is small compared to runtime. Finishing order of simulators depends. It may not necessarily be the same as starting order as simulation speed depends on the piece of the circuit model - some pieces are more complicated to simulate.

When all simulators are finished, then the Coordinator assembles sub results into final result and stores in the database. Results are passed to user when requested.

7 Experimental Results

In experiments we measured overall speedup, communication overhead and determined how well is the solution with current task partitioning scalable when number of processing units increases. Simulation was carried out on UltraSPARC IV+ 1500Mhz servers. Tomcat servlet engine and MySQL database were running on two core AMD Athlon 64 6000+ 3 GHz processor with 2 Gb memory. User applet was also running on the similar Athlon machine. Experiments show, that on the user computer circuit file loading takes about second for the over 3 Mb size circuit files depending on computer's performance. File transfer to the database and user notification takes about 6 seconds. Thereafter, simulation agent receives circuit file from the Coordinator 3-4 seconds later. Total communication delay was 12-15 seconds in case of distributed web-based solution. The total communication overhead was about 1% compared to single processor solution in case of largest circuits. The overhead depends on the size of the circuit and the number of test vectors simulated.

Table 1 Distributed simulation results

circuit	B17C	B18C	B21C	B22C
Max model partitions	13	8	12	13
Max model build time,s	0.24	1.83	0.32	0.37
Max subtask simul time, s	214	1534	146	195
Model size reduction, times	4.1	2.8	2.5	2.6
Speedup by model partit.	3.2	6.4	2.5	2.9
Speedup by test partitioning	10.3	7.9	8.7	10.0

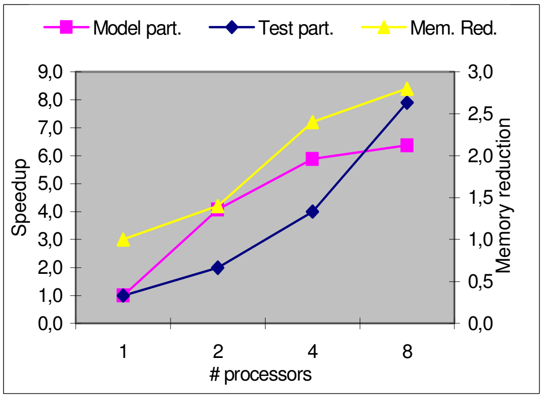


Fig. 3 Circuit b18 experimental results

Simulation results for sample circuits [12] are presented in Table 1. 100K test patterns were applied to each circuit. We can see that model build time for subtask (circuit slice) is very small (0.1% for b18 circuit) compared to simulation time. Final simulation time is dominated by longest subtask simulation time. Figure 3 illustrates results for largest (104580 logic gates) B18 circuit. It is interesting to see that model partitioning speedup is quite close to test partitioning speedup. Initially, up to 6 processors model partitioning has an advantage compared to test partitioning. Figure 4

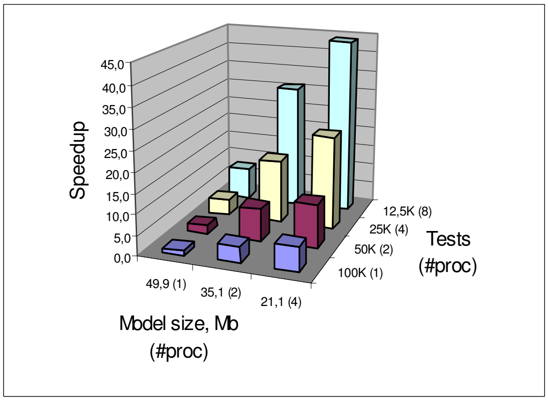


Fig. 4 Integrated speedup for circuit b18

shows ultimate speedup when model partitioning was combined with test set partitioning. In parenthesis, CPU count is given. In experiments we increased CPUs number on both axes i.e. we reduced model size and also vector count per CPU. Here we see that model was divided into 4 pieces and at the same time number of test vectors simulated on CPU is reduced from 100K to 6.25 (corresponding to 32 CPUs) resulting 45 time speedup. Optimal tradeoff between speed and processor count would be using 8 processors (2 processors for model partitioning dimension and 4 processors for test partitioning dimension). This would lead 15.6 time integrated speedup which is considerably better than just using model partitioning (4.1x speedup) or test partitioning (7.9x speedup).

8 Conclusions

We have presented how distributed computing paradigm can be used collaboratively in the field of digital fault simulation. WEB-based fault simulation solution allowing working through firewalls was described. We introduced credit based priority management, addressed task scheduling problem and improved task partitioning. In experiments, total communication delay for end user was approximately 12-15 seconds in average compared to local single processor solution. Communication overhead depends on the size of the circuit and the number of test vectors simulated. In practice, circuits and vector sets may be much larger overhead would be consequently smaller. Model partitioning reduced required memory amount in experiments up to 4 times. Proposed collaborative computing approach allows to solve large fault simulation problems which otherwise would be difficult to achieve due to memory and time constraints. Combining model partitioning with test partitioning allowed to achieve considerable speedups. Using 8 processors 15.6 time speedup was gained and maximum 45 time speedup was obtained with 32 processors.

Model partitioning algorithm could be possibly improved in the future to support finer granularity. However, combined with test set partitioning, it might be sufficient for moderately sized collaborative consortium, since in reality there is usually more than single circuit model in use at given time, consequently total number of tasks to be scheduled may be larger and appropriate for total number computers available. In the future also tradeoff estimation between model partitioning and test set partitioning could be introduced.

Acknowledgements. The work has been supported by Estonian SF grant 7483, FP7 REGPOT project CREDES, and Research Centre CEBE funded by EU Structural Funds.

References

1. MOSCITO, `http://www.eas.iis.fhg.de/solutions/moscito`
2. Schneider, A., et al.: Internet-based Collaborative Test Generation with MOSCITO. In: Proceedings of DATE Conference, Paris, France, March 4-8, pp. 221–226 (2002)
3. Ivask, E., et al.: WEB-Based Environment: Remote Use of Digital Electronics Test Tools. In: Virtual Enterprises and Collaborative Networks, pp. 435–442. Kluwer, Dordrecht (2004)
4. BOINC, `http://boinc.berkeley.edu/`
5. Teo, Y.M., Low, S.C., Tay, S.C., Gozali, J.P.: Distributed Geo-rectification of Satellite Images using Grid Computing. In: Proc. IEEE Int. Parallel & Distributed Processing Symposium (2003)
6. Devadze, S., et al.: Parallel Exact Critical Path Tracing Fault Simulation with Reduced Memory Requirements. In: 4th IEEE DTIS Conference, Cairo, Egypt, April 6-7, pp. 155–160 (2009)
7. Hwang, K., Xu, Z.: Scalable Parallel Computing: Technology, Architecture, Programming. McGraw-Hill, San Francisco (1998)
8. Knuth, D.E.: Fundamental algorithms, 2nd edn., vol. 1. Addison-Wesley, Reading (1973)
9. Eager, D.L., Lazowska, E.D., Zahorjan: Adaptive load sharing in homogenious distributed systems. IEEE Transactions on Software Engineering SE-12(5), 662–675 (1986)
10. Shivaratri, N.G., Krueger, P., Singhal, M.: Load distributing for locally distributed systems. IEEE Computer 25(12) (December 1992)
11. Condor: `http://www.cs.wisc.edu/condor/`
12. Example circuits: `http://www.cad.polito.it/tools/itc99.html`

Go4Flex: Goal-Oriented Process Modelling

Lars Braubach, Alexander Pokahr, Kai Jander,
Winfried Lamersdorf, and Birgit Burmeister

Abstract. Many companies consider business process management strategies a fundamental source for successful business operation. Despite this importance of business processes a conceptual and operational gap still exists between the business and the IT view of processes. In this paper we argue that an important reason for this gap is the strong focus of IT on the behaviour and execution perspective of workflows while more abstract and higher-level process properties are often neglected. This is especially apparent in the way processes are modelled and described on the IT-side using state of the art modelling approaches like BPMN. The presented Go4Flex research project, which is conducted in cooperation with Daimler AG, has the objective of bringing together both sides by establishing higher-level modelling concepts for workflows, which results both in increased intelligibility of workflow descriptions for business people and greater consideration for the way processes are described on the business side. The core idea of the approach is to strengthen the context perspective of a workflow by introducing different kinds of goals and goal relationships in addition to the established activity-centred behaviour model. The applicability of the approach is further illustrated with an example workflow from Daimler AG.

1 Introduction

Business processes form a challenging research area, in which the business and IT sides have to be conceptually and methodically aligned in order to

Lars Braubach · Alexander Pokahr · Kai Jander · Winfried Lamersdorf
Distributed Systems and Information Systems Group,
University of Hamburg, Germany
e-mail: `{braubach,pokahr,jander}@informatik.uni-hamburg.de`

Birgit Burmeister
Daimler Group Research, Sindelfingen, Germany
e-mail: `birgit.burmeister@daimler.com`

achieve their full potential [8]. The business side typically focuses on the elicitation of business processes, their management and controlling as well as their optimization while the IT side has to deal with their simulation, execution as well as real-time monitoring. Currently, the artefacts produced from methods at the business side are only partially adopted by IT so that a conceptual gap still exists. Few approaches have tried to address business process management (BPM) at a holistic level, which is a prerequisite for effective and efficient BPM. One example for such a holistic approach is the ARIS house of business engineering [18], also used at Daimler AG.

In practice, experience has shown that modelling means offered by ARIS, focussing on event process chains (EPCs), are insufficient for some processes at Daimler AG. While processes have been documented with EPCs, they have not been directly adopted by the workflow participants. One major issue is the strong focus on activities and their ordering, found in nearly all modelling languages including BPMN (business process modelling notation) [12]. As the processes considered at Daimler AG can frequently change, the abstractness of the process descriptions is essential for their long-term usefulness.

In this paper, we propose an approach based on the notion of process goals. The use of process goals aims at achieving a higher degree of abstractness in the process models by employing goals to describe what is to be achieved instead of how it should be done. The means for achieving a goal can then be described on a finer grained level using traditional activity oriented workflow languages. In addition, as goals play an important conceptual role on the business side, e.g. in well known process elicitation methods like business score cards [8], policy deployment [8] and with respect to process monitoring and evaluation metrics (key performance indicators, KPIs), they form an ideal basis for enhancing the integration level between both.

The next section will discuss related work with respect to workflow modelling approaches. Thereafter, in Section 3, the goal-oriented modelling approach will be presented in the context of the Go4Flex project. Section 4 further illustrates the approach by explaining selected concepts using an example workflow from Daimler AG and finally, Section 5 concludes the paper and highlights aspects of current and future work.

2 Related Work

In the literature many different description languages for workflows can be found. They can be coarsely divided into domain centred modelling approaches and execution oriented approaches. Example of the former group are e.g. EPCs [18], BPMN [12] and YAWL (yet another workflow language) [20], and of the latter group especially BPEL (business process execution language) [13], petri nets [15] and also rule based approaches like ECA (event condition action) [9]. The difference between both kinds of approaches is mainly the level of details supported by the language. Execution languages allow specifying more technical details and can thus be used to describe workflows that can be automatically executed. Yet, there is no fixed border between both, because on the one hand some of the more technical languages also offer comprehensible modelling concepts and on the other hand also mechanisms have been devised for converting a modelling language to an executable counterpart, e.g. from BPMN to BPEL [14] or from EPCs to rules [9].

To assess the usefulness of these approaches it is necessary to understand the different perspectives that are relevant during workflow modelling. Based on the early work of Curtis et al. [5], List and Korherr [10] proposed a holistic view on business processes consisting of five perspectives. The *functional view* focuses on the concrete execution of tasks and typically considers concepts for describing subprocesses and atomic tasks. By using the related *behaviour view* the sequencing of these elements is controlled. In many cases workflow control patterns like sequence, AND- or XOR splits and joins [19] are used for that purpose. The *informational view* is related to the process related data elements that are on the one hand necessary for task processing and on the other also produced by tasks. These data elements represent simple information as well as complex objects and business products or services and are often called process resources. In the *organizational perspective* it is highlighted who processes the tasks and how the distribution of tasks can be handled. Hence, in this view especially concepts like actors, roles and organizational units play an important role. Finally, the *context perspective* adds an overview or meta perspective to the process, which may contain important process characteristics like the process goals and their performance metrics in form of e.g. key performance indicators (KPIs).

Please note that the first four perspectives are quite commonly agreed upon and also build the fundamental building block of the ARIS house of business engineering [18]. New is the recent addition of the context perspective, which is until now not adequately reflected and connected to the concepts of the other perspectives. Today, most modelling approaches still focus too much on the functional and behaviour perspective, which can e.g. be observed in the BPMN, YAWL and BPEL languages. Organizational aspects are often reduced to the issue of task distribution and for the informational perspective established techniques like the entity relationship model (ERM) are utilized. An exception is the holistic ARIS approach, which pinpoints the right direction but is very heavyweight and offers dozens of modelling diagrams failing to achieve minimality in the sense of the parsimony principle [11] (introduce only as many concepts as really needed). Additionally, at the heart of the approach is the EPC behaviour description, which is a fine-grained task and event based modelling approach without goals. An interesting alternative to traditional workflow notations consists in omitting an explicit workflow description and building up workflows from basic activities with pre- and postcondition at runtime by using a planning algorithm as e.g. proposed in [6]. Such approaches help increasing the process agility but loose to some extent comprehensibility especially with repect to the business perspective.

In most aforementioned approaches, the explicit modelling of process goals and the relation of goals to the context as well as to the behaviour view is missing. Yet, goals are essential for understanding the rationale behind a process and also for directly steering the process execution. In addition the performance metrics, often using KPIs, form another important brick of the context perspective that currently gains much practical attention through the advent of real-time business activity monitoring (BAM) tools. Therefore, several BPMN tools try to offer solutions for the KPI modelling and process linking but fail altogether to link the KPI results to the underlying process goals, simply because they are not represented. Similar to our research objective is the approach of [17], who use the user requirements notation (URN) together with use case maps (UCM) and the goal-oriented requirements language (GRL) for modelling the context perspective. Also the idea of goal, KPI and process linking has been pushed forward in their approach. One main difference is that we use goals for functional and non-functional process aspects and hence make goals an integral conceptual element for the context and the behaviour perspective. i.e. the goal modelling is considered as starting point for the process definitions according to the goal-context method developed by Daimler AG [3] and also implemented in a commercial tool [4].

3 Go4Flex Approach

The Go4Flex project aims at providing advanced conceptual and software technical means for modelling and executing complex business processes. Go4Flex advocates a unifying approach that integrates the five aforementioned dimensions of workflows into a consistent framework. This framework is based on concepts, which have been developed in the area of agents and multi-agent systems. Existing multi-agent systems concepts cover large parts of the perspectives relevant for workflows and additionally are based on the unifying metaphor of (software) agents. The main research question of the DFG-funded Go4Flex project is to isolate interesting multi-agent ideas and make them usable also for workflows. Most importantly, Go4Flex focuses on the behaviour and context perspectives, which suffer among other things from their low conceptual connectivity. Thus Go4Flex is based on the notion of declarative goals [2, 21]. On the one hand goals allow capturing the reason for executing process activities. This facilitates taking a top-down perspective on processes that starts from high-level business goals instead of having to focus on low-level activities. On the other hand goals are an ideal concept for understanding the process context (why something is done and how good it is done). Thus we envision goals as one fundamental conceptual entity for both perspectives. Besides goals, the approach is conceived to integrate well with established concepts, e.g. by reusing available BPM concepts and techniques.

3.1 Concepts

Regarding the behaviour perspective, the Go4Flex approach is based on the goal-context method developed by Daimler Group Research [3]. This method advocates to model processes by starting from high-level goals. The process is refined by decomposing goals into a goal hierarchy. For each goal, the designer can specify a number of declarative properties, e.g. normal conditions to be fulfilled for a goal to be regarded as achieved or exceptional conditions, denoting when a goal should not be pursued. Once the goal hierarchy reaches an appropriate level of detail, the designer can start modelling plans to fulfil the leaf goals of the goal hierarchy. Plans represent courses of action that capture one of potentially many alternatives for fulfilling a superordinated goal. During process execution, the workflow engine automatically selects appropriate plans for each goal based on conditions that state the applicability of each plan to a given situation. Moreover, if a plan fails, the engine can try out other plans to achieve the same goal, if alternative plans are available.

Therefore, the method proposes two abstraction levels for process modelling: At the *goal level*, the process is modelled in a declarative way as a tree structure of goals to be fulfilled for the process to be successful. At the *plan level*, concrete activities for handling specific goals are described. For the goal level, a new notation is introduced. The plan level uses the existing BPMN as modelling language. The goal hierarchy represents the declarative properties of the process (conditions to be fulfilled), while the plans capture procedural aspects (sequences of actions to be executed).

The representation and execution semantics for the goal level has been directly adapted from the notion of goals in mentalistic belief-desire-intention (BDI) agents [2][1]. In order to simplify the goal semantics different functional goal kinds have been proposed, from which the most important ones are *achieve*, *maintain*, *query* and *perform* goals. These goal kinds map to different application use cases and thus help in naturally modelling the problem at hand. Achievement goals are the most common goal kind, which aim at the establishment of a user defined world state (declaratively expressed as a target condition). The goal executes plans until its condition is met or no more plans are available. On the other hand, a maintenance goal can be used to ensure that a specific world state is preserved. In case of a violation the goal tries to re-establish the condition by executing as many plans as needed. A query goal is used for information retrieval and only executes plans when the requested piece of information is not already available in the process context. Finally, perform goals are the simplest form of goals, which have a procedural semantics and are directly connected to executing possibly several plans. A detailed description of these goal kinds can be found in [2].

With respect to the context perspective our approach is inspired by the ideas of [17], who propose to use non-functional (soft) goals for expressing process performance metrics. In contrast to their exclusive treatment of non-functional goals, we envision that they should be part of the functional goal

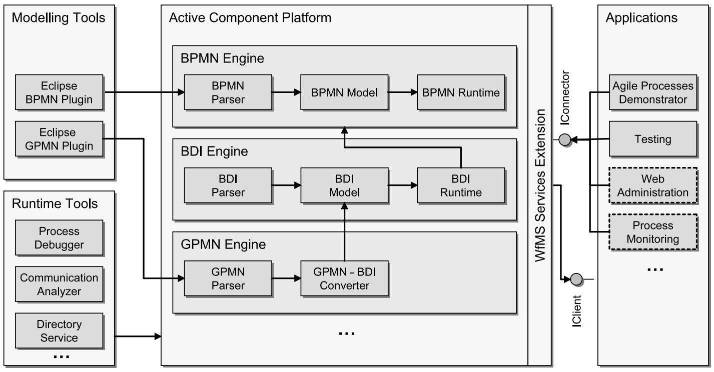


Fig. 1 Go4Flex architecture

hierarchy similar as in the agent methodology Tropos [7]. The connection between both is described with contribution links, which means that a functional or non-functional element either positively or negatively contributes to the non-functional goal. On lower levels non-functional goals can be decomposed into KPIs, which exactly determine the metrics of the non-functional goal and how it is measured. The explicit relationships between functional and non-functional process elements facilitate the way measurement results are interpreted, because their context is apparent.

Taken together the described approaches have several advantages compared to traditional techniques. First, the understandability of workflows is fostered by introducing goals as conceptual entities abstracting away from purely procedural descriptions (unlike e.g. BPMN). Second, the process description naturally facilitates design and runtime agility. Runtime agility is achieved by executing different plans for the same goal, depending on the current situation. Design agility results from that fact that the process descriptions are open to future changes and extensions. E.g. a process designer can add another plan for an alternative course of action without changing any of the existing process descriptions. Such an incremental process development is a very natural approach, because business users often prefer thinking in scenarios and not in complete processes, that include every potential alternative. Third, the conceptual integration of the context with the behaviour perspective will allow meaningful business process monitoring, because the metrics and KPIs are directly related to process goals.

3.2 Implementation

A number of necessary tools for modelling and execution of goal-based processes have been developed by the Go4Flex project. Figure 1 shows four key areas for tool support (modelling tools, runtime tools, platform, and applications). The starting point for creating new workflows are the modelling tools. Two editors are provided for developing new workflow models (see Figure 2, left). A graphical editor provides a special notation called GPMN (Goal-oriented Process Modelling Notation) which allows users to create Go4Flex goal hierarchies. This GPMN editor is also used to define the context of the workflow. The BPMN editor is used to create BPMN workflow fragments (i.e. plans), but can also be used to develop standalone BPMN workflows.

The workflow models generated by the editors can be used to instantiate workflows using the Jadex Active Component Platform [16]. The platform includes an interpreter for BPMN workflows or workflow fragments. GPMN workflows are supported by using a converter, which translates GPMN models into BDI agent models that can be executed by the Jadex BDI runtime.

Runtime tools (see Figure 2, right) provide diagnostics for running workflows. This includes a process debugger, allowing introspection, break points and stepped execution of workflows, a communication analyzer enabling the user to monitor and record messages passed between workflows and a directory service for deploying new workflow models at runtime.

User applications interact with the system from the perspective of a workflow participant. Currently, a GUI-based workflow client application and an automated workflow testing tool are available. Future additions could include a web-based administration tool and process monitoring tools. User

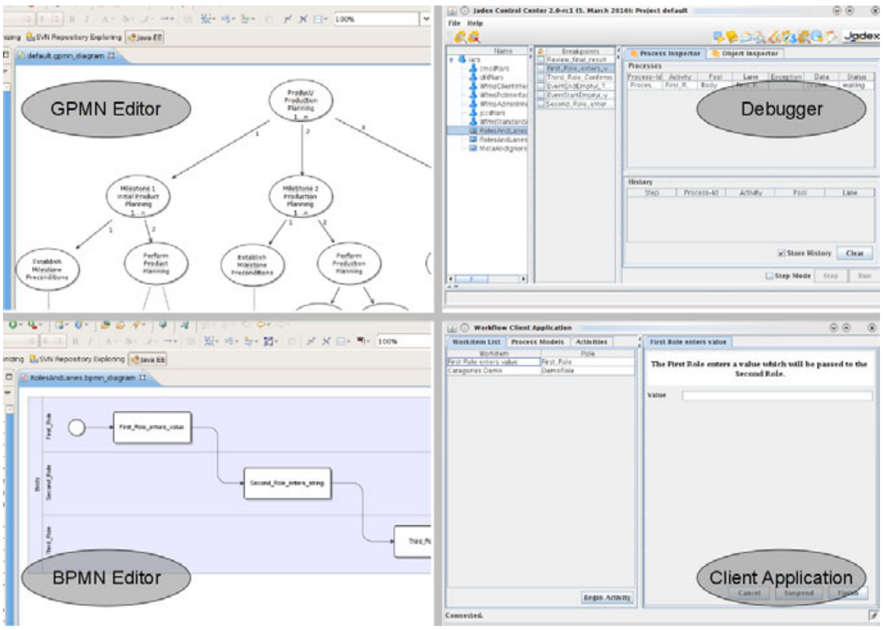

GPMN Editor
Debugger
BPMN Editor
Client Application
Jadex Control Center 2.0-rc1 (5. March 2010): Project default
Workflow Client Application
Product Production Planning
Milestone 1 Initial Product Planning
Milestone 2 Production Planning
Establish Milestone Preconditions
Perform Product Planning
Perform Production Planning
First_Role_enters_value
Second_Role_enters_string
The First Role enters a value which will be passed to the Second Role.

Fig. 2 Tools implemented for use in Go4Flex

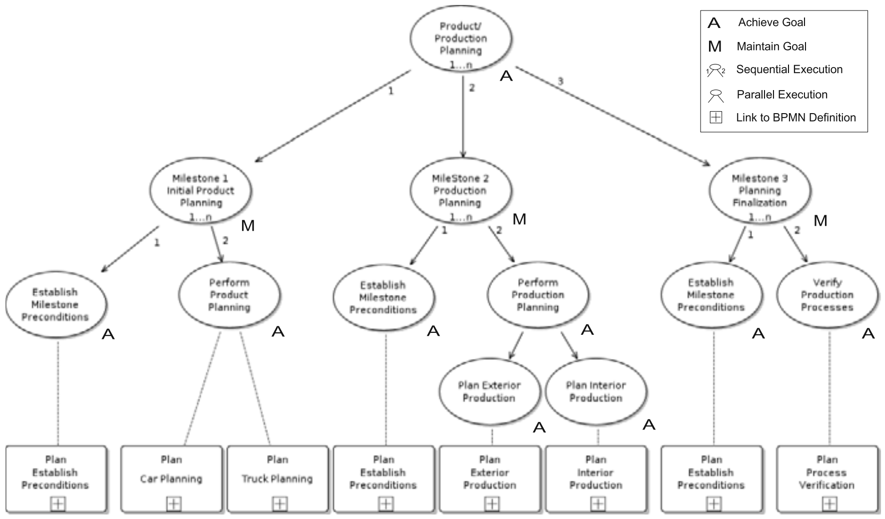

Product/ Production Planning 1...n
A
Achieve Goal
M
Maintain Goal
Sequential Execution
Parallel Execution
Link to BPMN Definition
Milestone 1 Initial Product Planning 1...n
MileStone 2 Production Planning 1...n
Milestone 3 Planning Finalization 1...n
Establish Milestone Preconditions
Perform Product Planning
Perform Production Planning
Verify Production Processes
Plan Exterior Production
Plan Interior Production
Plan Establish Preconditions
Plan Car Planning
Plan Truck Planning
Plan Exterior Production
Plan Interior Production
Plan Process Verification

Fig. 3 Goal hierarchy structure as used in the digital production workflow

applications interact with the runtime environment through a workflow management system services extension of the active component platform, adding additional runtime features like role management and work item handling.

4 Example Workflow

The Go4Flex goal concept has been used in several different areas at Daimler AG. The simplified digital production planning workflow shown in Figure 3 exemplifies how the new approach can be used in practice. The root of the goal hierarchy is an achieve goal which represents the general objective of the workflow.[1] The second layer of subgoals represents process milestones. The first milestone is the product planning stage, followed by planning the production processes of the product as the second milestone. The third milestone verifies the processes and finalizes the planning. The top goal is sequential, the next milestone goal only starts once the previous milestone has finished.

Employing milestones as the second layer of the workflow allows quick assessment of workflow progress by following the active milestone goal. For example, if the production planning milestone goal is active, the product planning stage has already finished. As alternative the second layer of goals can be structured by the area of operation and the milestone steps are delegated as subgoals further down the hierarchy. While this would be a reasonable design, it lacks the monitoring advantage of the former approach.

[1] The original workflow has been made abstract due to business secrecy reasons.

Instead, milestone subgoals have their subgoals tied to areas of operation. If possible, they can be executed in parallel. For example, planning the fittings and equipment of the interior can be done independently of exterior features like colour. If this is not viable, the goal can be declared sequential.

Once the hierarchy approaches a sufficiently fine-grained level, plans can be added to the goals. Plans represent BPMN workflow fragments which describe actions necessary to reach the goal. Since the goal hierarchy already decomposed the complex workflow goal into more basic goals, BPMN plans can be relatively simple, like requesting and storing information.

Milestone goals are modelled as maintain goals, monitoring whether the state of the context diverges from what the milestone is supposed to establish. In this case, the product planning goal monitors whether the conditions for a planned product are met. Once the top goal activates it, the context implies that product planning is necessary, activating the subgoals in the subtree. When finished, the top level goal activates the production phase goal.

However, the previous milestone goal is still actively monitoring the context. If the production planning invalidates assumptions made during the product planning, the product planning goal reactivates. Context conditions pause later milestone goals while the product planning milestone goal attempts to re-establish a product plan. Even though the whole subtree reactivates, subgoals that are leaf goals (i.e. goals without further subgoals) contain target conditions specifying their required context state. If the condition is already met, the goal finishes without plan execution. As a result, only unfulfilled subgoals will execute their respective plans a second time.

This illustrates how the Go4Flex approach allows flexible workflows. If a planning error invalidates work done in a previous milestone, the milestone reactivates and solves the problem without restarting the workflow. Thus the workflow reacts to changes in the context and maintains a consistent state.

5 Conclusion

In this paper it has been argued that a conceptual gap between the business and IT side of business processes still exists due to the many perspectives that have to be considered. In the IT area there is still a strong tendency to overweigh the importance of the behaviour perspective, which is clearly reflected in the state of the art modelling approaches like BPMN. In contrast, the context perspective, which represents a business near meta view on the reasons for process execution and its metrics, has not been subject of extensive technical research and is thus poorly supported from IT side.

The Go4Flex project targets this research strand by exploiting concepts from multi-agent systems for improving the conceptual underpinnings especially in the context perspective and its relation to the behaviour view. One fundamental building block of Go4Flex is the usage of declarative goals, which increase the understandability and abstractness of workflow descriptions. On the one hand functional goals help representing the underlying reasons for executing a process (enhancement of the behaviour view) and on the other hand non-functional goals support the understanding of the process metrics (enhancement of the relation between context and behaviour view).

In order to explore the new concepts workflow modelling, execution and management systems have been developed. The execution engine is currently capable of executing goal-oriented (GPMN) as well as BPMN workflows. The system is based on the Jadex active components platform, which is a generic infrastructure for running heterogeneous active elements and provides a suite of runtime tools for their management like debugging and conversation analysis. The engine for GPMN processes reuses an available BDI agent engine by converting the goal model to a BDI agent representation. To show the practical usefulness of the approach (an abstract version of) an example workflow from our project partner Daimler has been presented.

Future work will further develop and systematize the available modelling concepts. One aspect in this direction is the description of often recurring modelling fragments in form of goal-oriented workflow patterns. Furthermore the integration of non-functional goals represents an important step in the direction of goal-oriented process monitoring and improvement.

Acknowledgements. We would like to thank the DFG for supporting the technology transfer project Go4Flex.

References

1. Braubach, L., Pokahr, A.: Representing long-term and interest bdi goals. In: Proc. of (ProMAS-7). IFAAMAS Foundation, vol. 5, pp. 29–43 (2009)
2. Braubach, L., Pokahr, A., Moldt, D., Lamersdorf, W.: Goal Representation for BDI Agent Systems. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 44–65. Springer, Heidelberg (2005)
3. Burmeister, B., Arnold, M., Copaciu, F., Rimassa, G.: Bdi-agents for agile goal-oriented business processes. In: AAMAS 2008. IFAAMAS, pp. 37–44 (2008)
4. Calisti, M., Greenwood, D.: Goal-oriented autonomic process modeling and execution for next generation networks. In: van der Meer, S., Burgess, M., Denazis, S. (eds.) MACE 2008. LNCS, vol. 5276, pp. 38–49. Springer, Heidelberg (2008)
5. Curtis, B., Kellner, M., Over, J.: Process modeling. Commun. ACM 35(9), 75–90 (1992)
6. Ermolayev, V., Jentzsch, E., Karsayev, O., Keberle, N., Matzke, W.-E., Samoilov, V., Sohnius, R.: An agent-oriented model of a dynamic engineering design process. In: Kolp, M., Bresciani, P., Henderson-Sellers, B., Winikoff, M. (eds.) AOIS 2005. LNCS (LNAI), vol. 3529, pp. 168–183. Springer, Heidelberg (2006)
7. Giorgini, P., Kolp, M., Mylopoulos, J., Pistore, M.: The Tropos Methodology. In: Methodol. and Software Eng. for Agent Systems, pp. 89–106. Kluwer, Dordrecht (2004)

8. Sesselmann, W., Schmelzer, H.J.: Geschftsprozessmanagement in der Praxis. Hanser Fachbuchverlag (2008)
9. Knolmayer, G., Endl, R., Pfahrer, M.: Modeling processes and workflows by business rules. In: BPM, Models, Techniques, and Emp. Studies, Springer, Heidelberg (2000)
10. List, B., Korherr, B.: An evaluation of conceptual business process modelling languages. In: Proc. of SAC 2006, pp. 1532–1539. ACM, New York (2006)
11. Newell, A.: Unified Theories of Cognition. Harvard University Press, Cambridge (1990)
12. Object Management Group (OMG). Business Process Modeling Notation (BPMN) Specification, version 1.1 edition (February 2008)
13. Organization for the Advancement of Structured Information Standards (OASIS). WSPBEL Specification, version 2.0 edition (2007)
14. Ouyang, C., Dumas, M., ter Hofstede, A., van der Aalst, W.: From bpmn process models to bpel web services. In: Proc. of ICWS 2006, pp. 285–292. IEEE, Los Alamitos (2006)
15. Petri, C.A.: Kommunikation mit Automaten. PhD thesis, Institut fur Instrumentelle Mathematik, Bonn (1962)
16. Pokahr, A., Braubach, L., Jander, K.: Unifying agent and component concepts - jadex active components. In: Proc. of MATES 2010, Springer, Heidelberg (2010)
17. Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P., Weiss, M., Forster, A.: Business process management with the user requirements notation. Electronic Commerce Research 9(4), 269–316 (2009)
18. Scheer, A.-W., Nüttgens, M.: Aris architecture and reference models for business process management. In: Business Process Management, Springer, Heidelberg (2000)
19. van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases 14(1), 5–51 (2003)
20. van der Aalst, W.M.P., ter Hofstede, A.H.M.: Yawl: yet another workflow language. Information Systems 30(4), 245–275 (2005)
21. Winikoff, M., Padgham, L., Harland, J., Thangarajah, J.: Declarative & Procedural Goals in Intelligent Agent Systems. In: Proc. of KR 2002 (2002)

Towards a Reference Model for the Runtime-Phase of Semantic Data Mediation

Alexander Wöhrer, Siegfried Benkner, and Peter Brezany

Abstract. The concept of mediation has recently been recognized as an important component of service-oriented architectures. The overall process of mediation is commonly split into design-time, preparing the ground to enable semantic mediation and run-time, actually doing it. Typically, a mediator is seen as a coarse grained black box component hiding its internals. Our proposal defines the runtime phase of mediation as a dynamic process of a set of well defined components. The benefits of such a decomposition include component reuse and assembling of different combinations, development process improvement and conformance to the service paradigm. Our approach is evaluated in the context of the European @neurIST project.

1 Introduction

In order to combine data from different research domains and service providers for e-Science, various heterogeneities have to be taken into account. These include structural, syntactical, systemic and semantic heterogeneity [3]. The concept of mediation introduced by [19] has recently been recognized as an important component of service-oriented architectures (SoA) [5] to handle and resolve mismatches that prevent interoperability between its participants for multiple purposes. The overall process of mediation is commonly split into two distinct phases, namely design-time and run-time [9]. While the first phase prepares the ground to enable semantic mediation, the latter is concerned with actually doing it. Differences between the various approaches to provide semantics on the Web such as vocabularies, taxonomies, thesauri, etc. are explained in more detail in [16]. Ontology-based data

Alexander Wöhrer · Siegfried Benkner · Peter Brezany
University of Vienna, Nordbergstrasse 15/C/311,
A-1090 Vienna
e-mail: {woehrer,sigi,brezany}@par.univie.ac.at

integration systems have been studied for quite some time [10, 18]. The advantages of ontologies when used for data integration are discussed in more detail in [3]. One method to achieve linkage between semantics and resources is semantic annotation [15]. A big problem to overcome is the fact that databases work with a closed-world assumption, while ontology systems apply an open-world assumption [4]. Tailored ontology- and mapping languages have been proposed for this purpose in order to support very large quantities of data typically stored via relational database technology [11]. Due to the autonomy of participants and not preventable legacy in distributed systems, the application of semantic technologies should be non-intrusive and can therefore be seen as an additional layer, as pointed out in [17]. Since the advent of the Semantic Web, various efforts to enable semantic mediation in SoA have been reported [7, 14]. Typically, a mediator (a component performing the task of mediation) is seen as a coarse grained black box component. The internals, required workflows and used components are hidden and tightly bound. Additionally, the input as well as output format are predefined and often proprietary.

To allow the application and further improvement of semantic data mediation in distributed and ever evolving environments, like e-Science infrastructures, a systematic view on it is needed. We propose a reference model for the run-time phase of semantic mediation of data. We define this model as an abstract framework of a process for understanding significant entities in it and relationships between the entities. A reference model aims at providing common semantics, is used to support the development of specifications and tools and is not directly tied to any standards or technologies. Our model recognizes semantic data mediation as a dynamic process composed of several phases. The benefits of such a decomposition include component reuse and assembling of different combinations, development process improvement, and conformance to the service paradigm. The distribution of *"mediators"* itself - which we assume for their components as well - was already encouraged and motivated in [19] by greater economy of access, by better locality for maintenance and by issues of modularity.

2 Model Description

So far, research efforts in the area of semantic mediation in SoA have resulted in a rather coarse grained collection of components and specialized techniques, e.g. [6, 13], rather than a generic framework as comprehensive as in other areas of data access and integration, e.g. adaptive query processing or distributed query processing. As mentioned earlier, our reference model is based on decomposing the run-time phase of semantic data mediation into several distinct phases, namely acquisition, unification and transformation.

Components of the Model. The components of the reference model and their behavior is described as follows:

Acquisition: the acquisition phase is concerned with the transition of data to semantic information as well as providing means to expose/access semantically enriched data. A component *AC* of the acquisition phase acts as a source of semantic information *SI* in the specific model *m* from data source *D*. It can be seen as the input phase of the semantic data mediation task by providing semantic information (often by using the knowledge base (KB) to lift it).

Unification: the unification phase is concerned with the integration of semantic information. A component *UC* of the unification phase acts as a source of semantic information *SI* in the specific model *m* from semantic information *SI* in the same model. It can be seen as the processing phase of the semantic data mediation task by (1) combining semantic information in the same model from other components, therefore doing semantic integration and (2) applying semantic operations on some semantic information by using information from the KB, leaving it in the same model.

Transformation: the transformation component is concerned with providing the semantic information in the requested (semantic) target model. A component *TC* of the transformation phase acts as a source of semantic information *SI* in the specific model *t* produced by transformation of semantic information *SI* in the specific model *m*. It can be seen as the output phase of the semantic data mediation task by (1) transforming semantic information from one semantic model to another semantic model by using information from the KB, e.g. during context transformation and (2) transforming semantic information from one semantic model to another model (sometimes using information from the KB) or output/transportation format, which is typically called lowering (in contrast to lifting in the acquisition component).

We can now define our model for semantic data mediation *SDM* as

$$SDM = (AC_i, UC_j, TC_k)\ i \geq 1 \wedge (j+k) \geq 1$$

The components of the reference model cooperate to perform semantic data mediation. The data resources provide the input data (typically structured or semi-structured data with some sort of logical schema) to the acquisition components. The KB contains the output of the design phase (where do we come from, where do we want to go, what tools can we apply to achieve that) needed for every semantic mediation phase, e.g. ontologies, transformation rules, output of ontology combination processes as well as semantic annotations from acquisition components to describe available data sources. The KB is typically used by all phases and associated components. The acquisition components use it to lift the input data to a certain representation of semantically enriched information, e.g. ontology instances. The unification components provide mechanisms to integrate the semantically enriched information to unified semantic information, e.g. identifying equivalents and joining semantic information. The transformation component provides means to transform the output of the previous component to a target (semantic) format, e.g. context transformation and instance transformation. There is no strict sequence on how the components interact. It becomes clear from the definition of the components that an acquisition component can also provide input to a transformation component and a transformation component can also provide input to a unification component.

Benefits of the Model. The construction of a reference model for the run-time phase of semantic data mediation has the following key benefits:

- *Component reuse.* A framework allows component reuse and enables assembling of different combinations. For example, a particular approach to acquisition can be used with different ways of combination.
- *Systematic development.* The adoption of a reference model by the developers of a semantic data mediation system makes the development more systematic, as one can focus on the internals of his components of the framework, just worrying about the local and remote interfaces.
- *Conforming to SoA paradigm.* The application of the design of a framework following our reference model focusing on the interface of cohesive, decoupled modules conforms to the SoA paradigm, which is suitable for advanced and distributed applications such as e-Science.
- *Generic features.* The reference model is generic in the sense of being mediation environment and technique independent while being capable of capturing many existing semantic data mediation techniques.

As such, enabling semantic data mediation involves the interconnection of acquisition techniques, means of unification and transformation approaches. Multiple mediation components may be used to implement the run-time phase of a single data mediation task.

3 Implementation in @neurIST

Clinical diagnosis and treatment of complex diseases is often compromised by lack of data or fragmentation of existing data as well as by the missing integration of associated analysis and simulation processes. The @neurIST project [1] is addressing these problems by developing an advanced IT infrastructure for the management of all processes linked to research, diagnosis and treatment development for complex and multi-factorial diseases.

The purpose of semantic mediation in the @neurIST project is to enable semantic access and integration to a distributed set of heterogeneous data sources via specifying the query based on concepts of the domain model [8]. The output of the currently supported data mediation process is either (1) a mapping schema [20] to setup an @neurIST Data Mediation Service for retrieving data represented by the target concepts; or (2) a basic SQL query (including join conditions but without any further constraints) for retrieving data from an @neurIST Data Mediation Service. Later on the application of transformation functions and support for constraints might be added by (1) improving the granularity of the annotation process and (2) providing means to apply semantically registered transformation functions to the query tree.

The design-phase of our semantic mediation process is represented by a domain model implemented as shared ontology and semantic annotation of data sources at the logical schema level.

Ontology. The @neurIST Ontology was created for the domain of cerebral aneurysm. It models the relevant conceptual entities from Clinical Medicine, Molecular Biology, Epidemiology, Simulation, Disease and Risk Factors in OWL. The content was developed with domain experts, using a process that has been described elsewhere [2]. The ontology containes around 3000 classes, 100 relationship types, 2000 textual definitions and 1500 UMLS mappings. An ontology browser is available at `http://ontology.aneurist.org`.

Semantic annotation. In order to enable semantic mediation, the participating data sources have to be annotated with concepts from the @neurIST ontology. For this purpose, an adapted version of [12] is used as Semantic Registry and Broker (SRB).

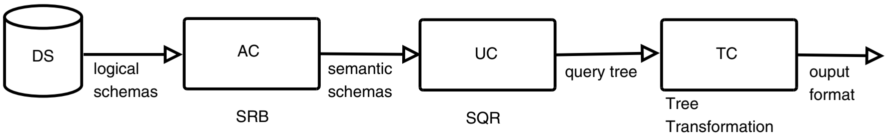


Fig. 1 A concrete instance of our reference model for semantic data mediation in @neurIST. Each component is implemented as a Web service.

Semantic mediation. The semantic mediation process in @neurIST is illustrated in Fig. 1. It uses a Semantic Query Resolver (SQR, our implementation of an *UC*), a Semantic Registry and Broker (SRB, an implementation of an *AC*) and a Tree Transformer (represents a *TC*). The SQR builds upon the SRB in the following manner: the SRB provides semantic schemes discovered and selected based on the required concepts while the SQR is responsible for combining the retrieved semantic schemes in the right manner, in our case building up an abstract query tree. The query tree is then transformed to the desired output format (either XML mapping scheme or textual SQL query statement) by the Tree Transformer component.

4 Conclusions and Future Work

The concept of mediation has recently been recognized as an important component of SoA to handle and resolve mismatches. Semantic technologies have a large impact on both design-time and run-time phases of mediation. Recent efforts to enable semantic mediation in SoA have focused on associated issues at a higher level, e.g. proposing new ontology languages and associated formalisms, in contrast to focusing on the process itself at a lower level. In this paper we propose a reference model for the run-time phase of semantic data mediation by decomposing it into sub-phases. We describe its benefits, constituent phases including their functionality and characteristics. The model has been evaluated by implementing a concrete instance of it in the context of @neurIST. Our future work agenda includes a more detailed and formal definition of it. Additionally, a basic set of components for each phase including a systematic analysis of their inputs and outputs as well as their granularities have to be elaborated.

Acknowledgements. This work was generated in the framework of the @neurIST Integrated Project, which is co-financed by the European Commission through the contract no. IST-027703.

References

1. Arbona, A., Benkner, S., Engelbrecht, G., Fingberg, J., Hofmann, M., Kumpf, K., Lonsdale, G., Wöhrer, A.: A service-oriented grid infrastructure for biomedical data and compute services. IEEE Transactions on NanoBioscience 6 (2007)
2. Boeker, M., Stenzhorn, H., Kumpf, K., Bijlenga, P., Schulz, S., Hanser, S.: The @neurist ontology of intracranial aneurysms: Providing terminological services for an integrated it infrastructure. In: Proceedings of the AMIA 2007 Annual Symposium (2007)
3. Buccella, A., Cechich, A., Brisaboa, N.R.: Ontology-based data integration methods - a framework for comparison. Colombian Journal of Computation, 450–456 (2005)
4. Cullot, N., Parent, C., Spaccapietra, S., Vangenot, C.: Ontologies: A contribution to the dl/db debate. In: SWDB 2003, pp. 109–129 (2003)
5. de Alencar Silva, P., Schiel, U., de Queiroz, J.E.R., Ribeiro, C.M.F.A.: Extending soa with semantic mediators. In: Proceedings of The 2006 IEEE/ACM International Conference on Signal-Image Technology and Internet-based Systems (2006)
6. Gannon, T., Madnick, S., Moulton, A., Siegel, M., Sabbouh, M., Zhu, H.: Semantic information integration in the large: Adaptability, extensibility, and scalability of the context mediation approach. MIT Sloan Working Paper 4541-05 (2005)
7. García-Sánchez, F., Valencia-García, R., Martínez-Béjar, R., Fernández-Breis, J.T.: An ontology, intelligent agent-based framework for the provision of semantic web services. Expert Systems with Applications 36(2), 3167–3187 (2009)
8. Kumpf, K., Wöhrer, A., Benkner, S., Engelbrecht, G., Fingberg, J.: A semantic mediation architecture for a clinical data grid. In: Grids Computing for Bioinformatics and Computational Biology, Wiley & Sons, Chichester (2007)
9. Mocan, A., Cimpian, E.: An ontology-based data mediation framework for semantic environments. International Journal on Semantic Web and Information Systems 3(2), 69–96 (2007)
10. Noy, N.F.: Semantic integration: a survey of ontology-based approaches. SIGMOD Rec. 33(4), 65–70 (2004)
11. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G.D., Lenzerini, M., Rosati, R.: Linking data to ontologies. Journal of Data Semantics 10, 133–173 (2008)
12. Qu, C., Zimmermann, F., Kumpf, K., Kamuzinzi, R., Ledent, V., Herzog, R.: Semantics-enabled service discovery framework in the SIMDAT pharma grid. IEEE Transactions on Information Technology in Biomedicine 12(2) (2008)
13. Scharffe, F.: Instance transformation for semantic data mediation. In: International Conference on Semantic Web and Web Services (2006)

14. Stollberg, M., Cimpian, E., Mocan, A., Fensel, D.: A semantic web mediation architecture. In: CSWWS 2006 (2005)
15. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation systems for knowledge management: A survey of requirements and state of the art (2005)
16. Uschold, M.: Where are the semantics in the semantic web? AI Magazine 24(3), 25–36 (2003)
17. Vetere, G., Lenzerini, M.: Models for semantic interoperability in service oriented architectures. IBM SYSTEMS JOURNAL 44(4), 887–903 (2005)
18. Wache, H., Voegele, T., Visser, T., Stuckenschmidt, H., Schuster, G., Neumann, H., Huebner, S.: Ontology-based integration of information - a survey of existing approaches (2001)
19. Wiederhold, G.: Mediators in the architecture of future information systems (1992)
20. Wöhrer, A., Brezany, P., Tjoa, A.M.: Novel mediator architectures for grid information systems. FGCS 21(1), 107–114 (2005)

Part IV
Distributed Algorithms and Complex Systems

A Distributed Algorithm for Personalized Trust Evaluation in Social Networks

V. Carchiolo, A. Longheu, M. Malgeri, and G. Mangioni

Abstract. This paper presents a distributed algorithm capturing the TrustWebRank metric, whose goal is to evaluate trust relationships among users in a social network. We first justify why a distributed version of TrustWebRank is useful and then we assess its performances, both in terms of complexity and bandwidth, through several simulations performed using an ad-hoc tool. The scenario we used for simulations is the Epinions.com social network. Results of simulations are finally discussed highlighting pros and cons of the proposed algorithm and possible approaches to improve its performances.

1 Introduction

Today, Web 2.0 and social-oriented applications are important research areas in the web applications world. They take advantage of both new technologies and new paradigm that enables a more sophisticated way to interact and exchange information as well as services. Main novelty in users' participation consists in the enhanced capability to collaborate in achieving some goals as, for instance, the buying and selling items online, or the exchange of interpersonal information. These collaborative networks are often referred to as virtual *social networks*, and it is possible to find a lot of examples of them in the Internet.

The spreading of virtual social networks and their impact on everyday activities enforces the role of reputation and recommender systems as a way of selecting *valid* (trusted) persons and/or *valid* (recommended) items.

The *recommendation* is an information filtering technique that generally exploits users' profiles to rank interesting items. Several approaches adopt an index of similarity between an user and the items he recommends, where the similarity depends on the user's direct knowledge about the items. Most important algorithms are

V. Carchiolo · A. Longheu · M. Malgeri · G. Mangioni
Dip. Ingegneria Informatica e delle Telecomunicazioni,
Facoltà di Ingegneria, Università degli Studi di Catania, Italy

referred as *collaborative filtering* [1, 2]. In this paper we focus on the recommendation based on trust relationship among users. Several authors [3, 4, 5, 6, 7, 8, 9] indeed proposed algorithms that associate each user with a trustworthiness index that is used to rank the items according to users' recommendation. The idea behind it is to select the items recommended by trusted users.

The authors of [3] and [10] highlighted the importance of *locality* with respect to *centrality*, since trustworthiness - in real world - depends on who is trusting who. The locality principle implies that each user may provide a different trust value about the same target (user). Other algorithms, as, for instance, Eigentrust [5], permit to assign a *global* value to each user based on the experience of all users in the network, according to the principle that if i trust j, and j trust k then also i may trust k taking into account the path joining i and j. We though that locality represents an important property that reproduces real social behaviour.

TrustWebRank[3, 10] is a metric based on the principle of locality that allows to calculate trust between any couple of users. The main contribution of this paper is to propose a *distributed* version of TrustWebRank that preserves the properties of the original algorithm.

However, the real applicability of such an algorithm depends on its performance both in terms of space (memory usage) and execution time. Since we are proposing an algorithm for distributed systems, we have to evaluate both the traffic generated across the network and the time needed to converge to significant values. The former characteristic aims at measuring the impact of the algorithm on the available bandwidth of the underlying transport network, whereas the latter measures the capability of the system to react quickly to modification of its status. This last feature is very important for social networks due to their dynamics, since the status of network changes often and quickly, so the algorithm should converge before the global system changes, i.e. before a large number of users changes their opinions (as in real network occurs).

To achieve significant results during the performance assessment we need a large dataset in order to highlight the problems connected with complexity: the dataset can be synthesized according to a given structure or it can be derived from a real social network. The use of a synthesized dataset allows to impose specific properties to the dataset - for instance, the network topology - but it is quite difficult to achieve a trust distribution faithful to real world. On the other hand, the use of a real dataset is usually more complex due to the difficulty to find an adequate example and to tailor it to the characteristics of our algorithm. Since we need a real distribution of trustworthiness indexes we must choose a real dataset, hence we used the Epinions.com dataset [11], a real recommender system; raw data we extracted by crawling the site have been further adapted since we need a *web of trust*, i.e. a social network where each node trusts others, whereas inside epinions users just provide recommendation about items. The resulting dataset was used in several simulations to assess the proposed algorithm.

The paper is organized as follows: next section briefly introduces TrustWebRank whilst section 3 presents the distributed version of TrustWebRank. The simulation results is presented in section 4 together with some discussions about its possible applications to social networks. Finally, section 5 presents conclusions and further works.

2 TrustWebRank

TrustWebRank [3, 10] is a *"personalized metric for trust"* that uses the principle of locality to construct the (possibly indirect) trust between agents i and j from existing direct trust among all agents; note that *direct* denotes the fact that i provided a trustworthiness to node j, whereas if i determines how much it should trust j by asking to other nodes (e.g. j's neighbours), this is denoted as *indirect* trust.

The metric is defined taking into account a scenario where several agents have trust relationship among each other, and agents could rate some *objects*. Usually each agent aims at knowing others' opinion about unknown objects and determining which are trustworthy. Of course, this implies that each agent has to evaluate also other's trustworthiness. Since in a social network an agent directly knows only few agents (*neighbours*), it has to evaluate the others in an *indirect* way, as stated before.

In this context, several proposals can be found in literature [12, 13]; one of the most important is *Eigentrust* [5] derived from *PageRank* [14, 15, 16] that is based on the concept of feedback centrality. The feedback centrality evaluates the centrality value of an agent according to the centrality value of its neighbours. All the algorithms based on PageRank aim at calculating a single - and global - trust value associated to each agent inside a network. In [3, 10] the authors pointed out a variety of criticisms against these approaches, the most important are:

- problems related to the uniqueness of the solution;
- lacks of combination of direct and indirect trust;
- the effect of the normalization of trust.

TrustWebRank rides above limits, proposing a method that is not based on the notion of centrality. In the following we briefly recall the main definitions needed to calculate the indirect trust among each couple of agents (for a deeper discussion on TrustWebRank please refer to [3, 10]).

Usually, the scenario is described by a graph where each agent is represented by a node, and the direct trust i to j is represented by the arc connecting them. More formally, we define the mentioned network as the graph $G(V,L,T)$, where V is the set of nodes, L is the set of oriented arcs, and T is the set of labels (i.e. trust values). Therefore, given $i,j \in V$, the oriented arc (i,j) means that i trusts j with a level $T_{ij} \in T$. We also define N_i as the set of the neighbors of the agent i, as $\forall j \in V : \exists (i,j) \in L$.

Let $T_{ij} \in T$ (defined in $[0,1]$) the trust matrix that describes the above defined network, where $T_{ij} = 0$ if $(i,j) \notin L$. Formula (1) represents the stochastic matrix S obtained by normalizing T.

$$S_{ij} = \frac{T_{ij}}{\sum_{k \in N_i} T_{ik}} \tag{1}$$

let us note that the normalization hides the concrete agent i's trust level to its neighbours, in fact either $T_{ij} = 0.01, \forall j \in N_i$ or $T_{ij} = 1.0, \forall j \in N_i$ results in the same S_{ij}! In the following we refer to this problem as *flattening*.

The trustworthiness of each agent is defined as follows:

$$\tilde{T}_{ij} = \sum_{k \in N_i} S_{ik} \cdot \tilde{T}_{kj} \tag{2}$$

where each element represents either the direct trust - whenever exists - or the indirect trust.

The formula in (2) solves the problem of centrality but does not take into account correctly direct experience.

Further step is to insert the direct experience, represented by S_{ij} obtaining the formula (3) that in matrix notation can be denoted as in formula (4)

$$\tilde{T}_{ij} = S_{ij} + \beta \sum_{k \in N_i} S_{ik} \cdot \tilde{T}_{kj} \quad \forall i,j \quad \text{where} \quad \beta \in [0,1) \tag{3}$$

$$\tilde{T} = S + \beta \cdot S \cdot \tilde{T} \tag{4}$$

Equation (4) requires the inversion of a matrix to be solved, that poses computation problems especially for large graphs. However, it is possible to solve the above equations thanks to the iterative method shown in equation (5).

$$\tilde{T}_{ij}^{(k+1)} = S_{ij} + \beta \sum_{l \in N_i} S_{il} \cdot \tilde{T}_{lj}^{(k)} \quad \text{where} \quad \tilde{T}_{lj}^{k} \quad \text{is computed at step} \quad k \tag{5}$$

It is possible to state that, under certain conditions, there exists a unique non-trivial solution to equation (4). The first term (S_{ij}) takes into account direct trust and, being $\beta < 1$, it prevails on indirect trust. Finally, the resulting matrix is not normalized avoiding the *flattening* problem.

Let us make some considerations about the complexity of the metric:

1. since the result is a matrix we need $|V| \cdot |V|$ cells, whilst Pagerank-based algorithms only need $|V|$ cells;
2. TrustWebRank authors [3, 10] show that the computing complexity is $O(N^2)$, where PageRank is $O(N)$. They also suggests that complexity could be reduced stopping the computation before reaching far ways agents, but does not provide any analysis about it;
3. the original formulation of TrustWebRank is not directly applicable to distributed systems, as for instance peer-to-peer networks.

3 A Distributed Implementation of TrustWebRank

In this section we propose a distributed implementation of TrustWebRank that comes from a reinterpretation of the equation 5 in a distributed context. In the proposed algorithm both evaluation and storage is distributed, each node i is responsible for computing and storing its undirected trust $\tilde{T}_{ij}$ towards node j. Such a computation is carried out by the node i exchanging messages with its neighbourhood. The distributed TrustWebRank algorithm is shown in Algorithm 1.

Algorithm 1 Distributed TrustWebRank Algorithm

```
each node $i \in V$ do:
  $k = 0$
  repeat
    Query all nodes $l \in N_i$ for $\tilde{T}_{lj}^{(k)}$
    Compute $\tilde{N}_i^{(k+1)}$ as $\{j \in V : \tilde{T}_{lj}^{(k)} \neq 0\} \bigcup N_i$
    Compute $\tilde{T}_{ij}^{(k+1)} = S_{ij} + \beta \sum_{l \in N_i} S_{il} \cdot \tilde{T}_{lj}^{(k)}$ where $j \in \tilde{N}_i^{(k+1)}$
    Compute $\delta = ||\tilde{T}_{ij}^{(k+1)} - \tilde{T}_{ij}^{(k)}||$
    $k = k + 1$
  until $\delta < \varepsilon$
```

For a sake of clarity, we define the *indirect neighbourhood* $\tilde{N}_i^{(k)}$ of a node i at step k as the union of its immediate neighbours N_i (see sec. 2) and the indirect neighbourhood $\tilde{N}_l^{(k-1)}$ at step $k-1$ of all nodes $l \in N_i$.

Note that due to the distributed nature of the algorithm, each node initially knows only its direct neighbours, so the $\tilde{N}_i^{(0)} \equiv N_i$. Therefore at first step a node is able to compute only the trust values of its adjacent nodes (i.e. $\tilde{T}_{ij}^{(1)} = S_{ij}$). As iterations are performed, a node can get knowledge of some other nodes far from it. Therefore, in general, the $\tilde{N}_i^{(k)}$ of a node i grows up as time pass and sometimes it can include all nodes in V, i.e $|\tilde{N}_i^{(k)}| = |V|$. As shown in the Algorithm 1, after the undirected neighbourhood computation, each node proceeds with the computation of the indirect trust towards those nodes included into the $\tilde{N}_i^{(k+1)}$.

The last step of the algorithm is the computation of the local residual δ, defined as $||\tilde{T}_{ij}^{(k+1)} - \tilde{T}_{ij}^{(k)}||$. A node is considered to be converged when the local residual δ becomes less than a given threshold ε. It is noteworthy that not all nodes converge at the same time, due to the local convergence criteria.

Figure 1 shows the first 3 iterations of the distributed TrustWebRank on a toy-directed graph made of 7 nodes and 11 edges labelled with random trust values $\in [0,1]$.

4 Experimental Results

In this section we present an example of application of the distributed version of TrustWebRank to the dataset obtained by crawling Epinions.com, that is a recommendation system that *"helps people make informed buying decisions"*[11].

Epinions.com achieves its goal through unbiased advice, personalized recommendations, and comparative shopping. Epinions.com allows registered *user*s to rate

A Distributed Associative Classification Algorithm

Djamila Mokeddem and Hafida Belbachir

Abstract. Associative classification algorithms have been successfully used to construct classification systems. The major strength of such techniques is that they are able to use the most accurate rules among an exhaustive list of class-association rules. This explains their good performance in general, but to the detriment of an expensive computing cost, inherited from association rules discovery algorithms. We address this issue by proposing a distributed methodology based on FP-growth algorithm. In a shared nothing architecture, subsets of classification rules are generated in parallel from several data partitions. An inter-processor communication is established in order to make global decisions. This exchange is made only in the first level of recursion, allowing each machine to subsequently process all its assigned tasks independently. The final classifier is built by a majority vote. This approach is illustrated by a detailed example, and an analysis of communication cost.

1 Introduction

Many of data mining algorithms are suitable for distributed computing, this paper focuses on classification algorithms based on association rules, a.k.a "Associative Classification algorithms" (AC) which is a promising approach in data mining that utilizes the association rules mining techniques to construct classification systems. Several studies [10][11][15] have provided evidence that such algorithms are able to extract classifiers competitive with those produced by many of traditional classifiers like decision trees, rule induction and probabilistic approaches. The advantage of this approach lies in (1) its simplicity compared to other mathematical classification methods, and its competitive results in term of accuracy (2) the exhaustive quest

Djamila Mokeddem · Hafida Belbachir
University of Sciences and Technologies Mohamed Boudiaf Oran,
Dept. of Computer Sciences, Laboratory of Signal Systems and Data LSSD,
B.P 1505 Elmnaouer Oran Algeria
e-mail: Mokdjamila@yahoo.com, H-belbach@yahoo.com

for all rules allow to find many interesting and useful rules not being discovered by the state-of-the-art classification systems like C4.5 [14] (3) It produces an understand model based on user-specified constraints. A typical associative classification system is constructed in two stages: (1) generating the complete set of class association rules (CARs) that satisfy the user-specified minimum support (called minSup) and minimum confidence (called minConf); (2) selecting the most accurate rules to build the classifier (pruning phase). The major strength of such systems is that they are able to use the most accurate rules for classification because their rule generators aim to find all rules. This explains their good performance in general. However, with very large datasets, both rule generation and rule selection in such systems are time consuming. A straightforward solution to this problem is surely a distributed and/or parallel computing which must achieve two principal goals: first improving the time processing of AC algorithms applied in very large databases, by a partitioning of data set, second maintaining a performance, in most cases, comparable to or better than learning a single classifier from the union of the distributed data sets.

This issue was not addressed much in the literature (cf. 3.3), but there are several studies on distributed association rules mining which may be adapted to a distributed classification context. In this paper, we propose a distributed algorithm for associative classification based on FP-growth algorithm [8], by adapting it to the classification task, and by exploiting its ”divide and conquer” strategy. The remainder of this paper is organised as follow: the next section gives an outline on AC algorithms, the section 3 provides a review of related works. In section 4, a detailed framework of the proposed distributed algorithm is described, following by a conclusion and discussion on the tracks which always deserve to be explored.

2 Associative Classification Algorithms

Classification is one task of data mining which allows predicting if a data instance is member of a predefined class. In a more general objective, the task of association rules discovery focus on finding rules of the form A→B relating disjoint set of database attributes, which is interpreted to mean “if the set of attribute-values, or “item-set” A is found together in a database record, then it is likely that the set B will be present also”. The discovering of “rules” in these two tasks is a strong common point which has inspired several researchers to explore the possibility of using association rules mining methods in classification.

The basic idea is to view classification rules as a special case of association rules, in which only the class attribute is considered in the rule’s consequent. In order to build classifier, an AC algorithm uses a training data set D to produce class association rules (CARs) in the form of X→y, where X⊆I, the set of the “items”, and y∈Y, the set of the class labels. An item is described as an attribute name X_i and a value x_i, denoted $< X_i, x_i >$. The rule R: X→y has support s, denoted as supp(R), if s% of the cases in D contain X and are labelled with the class y. A rule R: X→y holds in D with confidence c, denoted as conf(R), if c% of the cases in D that contain X are labelled with the class y. The main goal is to construct a set of CARs that satisfies the minimum support and minimum confidence constraints, and that is able to predict the classes of previously unseen data (the test data set), as accurately as possible. Other measures can be used like the conviction which take into account the class frequency, it is defined as : conv(X→y)=(1-supp(y))/(1-conf(X→y)), where supp(y) is the frequency of the class label y in D [3].

2.1 Generating Rules for Classification

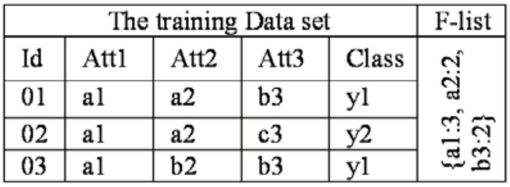
To find the complete set of the classification rules passing certain support and confidence thresholds, AC systems mine the training data set using a variant association rules discovery method. Several AC approaches, e.g. CBA [11], have been adopted from a priori algorithm [2] which is based on an iterative process: generate candidates, and generate frequent item sets. This method is computationally expensive because it needs repetitive scans of the data base. Others, like CMAR [10], are based on FP-growth algorithm [8], one of the most efficient serial algorithms to mine frequent item-sets. The effectiveness of FP-growth algorithm comes from its FP-tree structure which compresses a large database into a compact form, and its divide-and-conquer methodology. Furthermore, it avoids candidate generation, and needs only two scans of the entire data base.

2.2 Pruning Rules

Pruning techniques rely on the elimination of rules that are either redundant or misleading from taking any role in the prediction process of the test data objects. The removal of such rules can make the classification process more effective and accurate. For example, CBA [11] ranks all discovered rules by precedence ordering (using confidence then support), and uses the database coverage technique which guarantees that every classification rule can at least classify one training instance correctly (a rule r covers an instance d if d satisfies all conditions of the rule body of r). CMAR [10] takes all rules that apply within a confidence range. A survey of pruning methods is documented by [15].

3 Related Work

The design of distributed associative classification systems does not differ too much from the design of distributed association rules mining methods, at least in the first step (generation of CARs). For this reason, a set of recent works on distributed FP-growth algorithm will be presented. This approaches proposed for association rules discovery in distributed computing environment, may be adapted to generate the class-association rules in a distributed classification context.



The training Data set					F-list
Id	Att1	Att2	Att3	Class	{a1:3, a2:2, b3:2}
01	a1	a2	b3	y1	
02	a1	a2	c3	y2	
03	a1	b2	b3	y1	

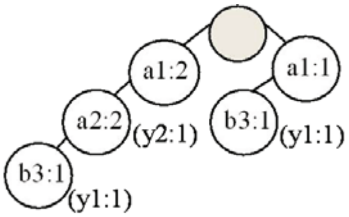


Fig. 1 An example of constructing FP-tree taking into account the class labels

3.1 *Class-Association Rules Mining Based on Serial FP-Growth*

In AC algorithms based on FP-growth, classification rules are generated by adapting this algorithm to take into account the class label. The key idea used in CMAR system [10] for example is to store the class label in the FP-tree structure. We illustrate this by the following example (fig.1), where minSupp=2 (66%). First, the training data set is scanned to find the list of frequent items F-list sorted by descending order of supports. A second scan is done to construct the FP-tree, where each frequent item in each record is inserted in the tree according to the order of F-list. This operation allows to represent a shared prefix of several records only once, like the prefix "a1a2" shared by the first and second records. Support counts are summed in each node, and the class label is attached to the last node in the path.

In the phase of frequent CARs mining, frequent items are recursively mined as follow: construct conditional pattern base (CPB) for each item in F-list, construct conditional FP-tree (CFP-tree) from CPB. When the conditional FP-tree contains a single path, the recursion is stopped and frequent patterns are enumerated using a combining method. This process is more detailed in [8]. Once a frequent item is found, rules containing this item can be generated immediately, by merging class values in the lower branches of considered item. For example, to find rules having b3, CPB is built by enumerating prefixes of b3 in the initial FP-tree:{a1a2(y1:1); a1(y1:1)}. Then CFP-tree is obtained considering only frequent items in CPB. In this case, CFP-tree contains only one node a1:2, and the CARs generated are: b3→y1, and b3a1→y1, obtained by combining the node a1 of CFP-tree, and the considered item b3. The two rules have support 2 and confidence 100%. The remaining rules can be mined similarly.

3.2 *Distributed Association Rules Mining Based on fp-Growth Algorithm*

Many studies have proven that the frequent item-set mining times of all algorithms for all data sets grow exponentially as the threshold increases. FP-growth algorithm is not an exception despite its improvements over a priori-based techniques. In very large databases and with low value of minimum support, the used FP-tree structure may not fit into the main memory. Several algorithms have been proposed in the literature to address the problem of frequent item-set mining in parallel and distributed environment. Many of them are based on a priori algorithm, e.g. CD, DD, and CaD strategies [1], but they still suffer from sequential a priori limitations. Inspired by its intrinsic divide-and-conquer nature, and its performance gain over a priori-based methods, FP-growth has been the subject of several studies in parallel and distributed frequent item-sets mining. The main goal is to reduce the time spent in computation, with a minimum of interaction between data sites. A compromise between memory constraint and inter-processor communication cost is difficult to obtain in distributed frequent item-set mining. The ideal state is a model that allows a "total" independent processing between data sites. This may be possible generally if a certain form of data replication is assumed like in [4][7]. Others works are based on an information exchange in different forms: local conditional pattern bases [12][13], sub-trees [9][18][17][6], or transactions [19]. Clearly there are tradeoffs between these two different approaches. Algorithm designers must compromise between approaches that copy local data globally and approaches which retrieve information from remote machines as needed. All these works divide the data set over several processors based on random horizontal partitioning method, and different solutions are proposed to more detailed issues such as load balancing strategies.

3.3 Distributed Associative Classification

To our knowledge, and up to the writing of these lines, there are been few published studies that have focused on distributed associative classification. Thakur (2008) [16] proposed a parallel model for CBA system [11]. The parallel CAR generation phase is an adaptation of CD approach [1] for mining CARs in associative classification. The training data set is partitioned among P processors, and in each iteration, each site calculates local counts of the same set of candidates and broadcasts these to all other processors. After a synchronization step, CARs are generated from frequent item-sets. This process is repeated until production of the complete set of CARs in each site. In the parallel classifier builder phase, the rules are pruned using "coverage database technique", applied to partitioned training data set. This strategy inherits two major limitations of CD approach : a tight synchronization at the end of each step, and the duplication of the entire set of candidates at each site. In addition, the data base coverage strategy applied in distributed setting is expensive in communication overhead.

4 The Proposed Model

The goal of this work is to propose a distributed model for associative classification technique. We start by presenting a simple sequential algorithm, based on FP-growth approach.

4.1 The Sequential Associative Classification Algorithm

In the sequential version (Algorithm 1), the list of CARs is generated by FP-growth algorithm like in CMAR [10]. To evaluate the rank of a rule, a total order is defined as follow: Given two rules r1 and r2, r1 is said having higher rank than r2, denoted as r1$\prec$r2, if and only if (1) $conv(r1) > conv(r2)$; (2) $conv(r1) = conv(r2)$ but $conf(r1) > conf(r2)$; (3) $conv(r1) = conv(r2)$, $conf(r1) = conf(r2)$ but $supp(r1) > supp(r2)$; or (4). The convictions, confidences and supports of r1 and r2 are the same, but r1 has fewer attribute values in its left hand side than r2 does. The use of conviction measure might give more information on the rule rank, because the frequency of class is taken into account (cf. 2).

Algorithm 1 Sequential AC

```
  Input: training data set D, minSup, minConf
  Output: the list of CARs
  1) Scan D and build F-list;
  2) Scan D and build FP-tree structure, storing also the class labels;
  3) For each item in F-list, construct CPB and CFP-tree;
  4) Mine recursively each CFP-tree, by extracting CARs;
  5) Sort the CARs according to the measures;
```

Seeing that this algorithm will be parallelized, we propose, as preliminary idea, to built the classifier by all the rules without any pruning operation. The classification of a new record will be made as follow: go through the list of all sorted rules until the first rule that covers the example is found; classify the example according to the class at the right -hand side of the rule. If no rule covers this example, mark the example as unclassified.

4.2 Associative Classification in a Distributed Environment

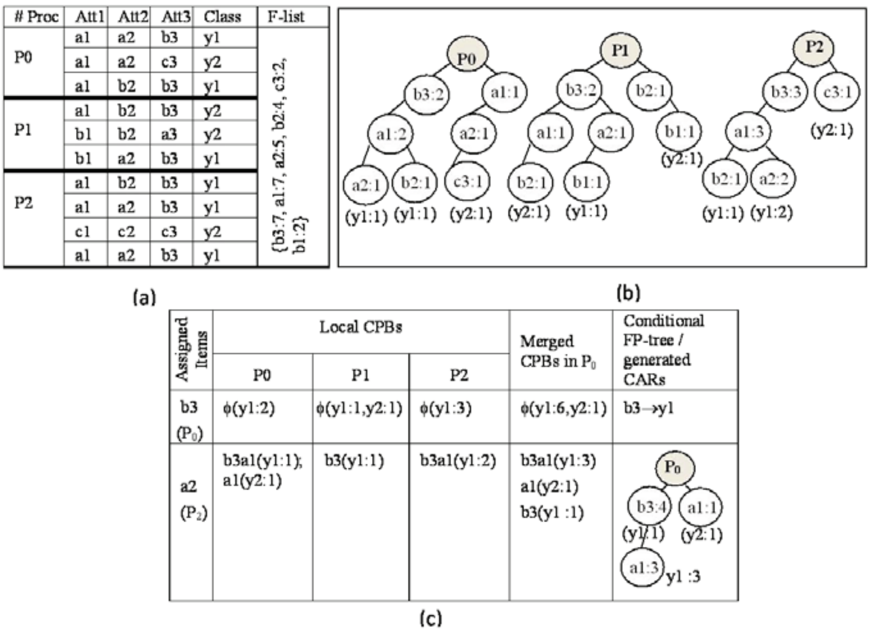
In this scheme (Algorithm 2), P processors work in parallel on a shared-nothing architecture. The processors begin with counting local support for each item. This counts are then exchanged across the group in order to each processor calculates their sum. After discarding globally infrequent items, each processor constructs the F-list structure sorted by descending order of frequent item supports (lines 2-4). In the next phase, local CFP-trees are built by scanning local data partitions and considering only local items belonging to F-list. The class label is attached to the last node in the path (line 5). Thereafter, the local CFP-trees are used in parallel to generate local conditional pattern bases CPBs in each processor, for each item in F-list (line 6).

Algorithm 2 Distributed AC

```
Input: P partitions $D_1$, $D_2$, ...$D_p$ of a training data set D; minSup, minConf
Output: A classifier $Cl_{maj}$ representing a majority vote between P classifiers
1) For j=0 to P-1 do in parallel, in processors $P_0..P_{p-1}$
2) Scan local data partition $D_j$ and count local support for each item i, $Supp_j(i)$;
3) Broadcast $Supp_j(i)$ for each item i;
4) Build F-list sorted in support descending order, by a global reduction $Supp(i) = \sum Supp_j(i)$, let k the size of F-list.
5) Scan local data partition and build local FP-tree with local items;
6) For each item i in F-list which belongs to the local partition, build local conditional pattern base $CPB_j(i)$, also including class values;
7) The k items of F-list are equitably assigned to each processor
8) Send $CPB_j(i)$ to the corresponding processors.
9) Merge received and local CPB for assigned items i;
10) Delete local FP-trees and local CPB for not assigned item i
11) Apply Sequential AC algorithm independently on assigned items;
12) Sort locally the subset of rules $CAR_j$;
13) End do in parallel
```

This partial information will be communicated between the processors and merged to generate the initial global CPB for each item. A trivial method to assign tasks to different processors is to perform a bloc-cyclic item distribution, so that the item i will be assigned to the processor number (i-1)%P. Thereby, each processor will send its local CPBs to corresponding processors avoiding to send "all" to "all" (line 7-9). This data distribution strategy could "contribute" to balance the load between the processors, because generally, the amount of work needed to process an item of F-list increases for the items with low supports, thus those having longer prefixes. After this communication, each machine independently mines recursively its assigned items, without any need of synchronization. At this level, several data structures can be deleted from distributed main memories: local CFP-trees and local CPBs for the items assigned to other processors (line 10). At the end, if each site products a subset of CARs, the union of all subsets must be exactly the total rule set obtained in the serial version of the algorithm.

To classify a new record d, a simple and intuitive technique consists in performing a majority vote between the P components of the final composite model. This strategy was used successfully in ensemble learning methods, with e.g. decision trees as base classifier [5]. The instance to classify is presented to each classifier Cl_j, (j= 1..P), like in the sequential version (cf. 4.1), thus it will be classified according to the prediction of the majority, denoted by $Cl_{maj}(d)$=$\text{argmax}_{c \in Y}\left(\sum_{j=1}^{P} I(Cl_j = c)\right)$, where c is a class label, Cl_j is the classifier obtained in the processor j, and $I(A)$ is an indicator function that returns 1 if A is true and 0 otherwise. The example in (fig.2) illustrates a part of this process. After communicating local counts, processors



# Proc	Att1	Att2	Att3	Class	F-list
P0	a1	a2	b3	y1	{b3:7, a1:7, a2:5, b2:4, c3:2, b1:2}
	a1	a2	c3	y2	
	a1	b2	b3	y1	
P1	a1	b2	b3	y2	
	b1	b2	a3	y2	
	b1	a2	b3	y1	
P2	a1	b2	b3	y1	
	a1	a2	b3	y1	
	c1	c2	c3	y2	
	a1	a2	b3	y1	

Assigned Items	Local CPBs			Merged CPBs in P0	Conditional FP-tree / generated CARs
	P0	P1	P2		
b3 (P0)	φ(y1:2)	φ(y1:1,y2:1)	φ(y1:3)	φ(y1:6,y2:1)	b3→y1
a2 (P2)	b3a1(y1:1); a1(y2:1)	b3(y1:1)	b3a1(y1:2)	b3a1(y1:3) a1(y2:1) b3(y1 :1)	P0 b3:4 a1:1 (y1:1) (y2:1) a1:3 y1 :3

Fig. 2 An example of generating CARs in a shared-nothing architecture

compute the sum and discard non global frequent items (fig.2.a). Instead of building a global FP-tree, which may not fit in main memory, local FP-trees are constructed in each processor, with the same method presented in figure 1, but considering only global frequent items, i.e. belonging to F-list (fig.2.b). Thereafter, local CPBs are built in each processor, for each item in F-list structure, and communicated to corresponding processors using a bloc-cyclic distribution strategy. So, the items {b3, b2} will be assigned to processor P_0; {a1, c3} to P_1; and {a2, b1} to P_2. The process of local CARs mining will be executed independently in the three processors resulting on the classifier $Cl_1 \cup Cl_2 \cup Cl_3$. For example (fig.2.c), in the first recursion level, processor P_0 generates the CAR "b3→y1" for the assigned item b3. On the other hand, processor P_0 repeats the recursion on the built CFP-tree until obtaining one branch.

5 Communication Cost Analysis

In the proposed algorithm, the information exchange allows to build the initial global F-list, and the initial CPBs. The first communication phase is not very expensive because each processor broadcasts only support counts for all items (q items) to all the processors (P processors). So in this stage there are P(P-1)q messages exchanged. All this information exchanged is integer valued and its volume is very small. On the other hand, the major communication cost comes from the exchange of the initial CPBs across all processors. This can be optimized by avoiding an all-to-all strategy. Each processor has information on what it must send to each other processor, such as the CPB for the item i (i=1..k) is sent to the processor number (i-1)%P. In this phase, each processor send (k/P) CPBs to corresponding processors, so the total number of messages is k. The larger of these messages depends on the maximum size of CPB. In the worst case, the item is present in all transactions of the data partition, with the largest prefix of size (m-1) (m is the number of attributes). So the maximum message size in this communication phase is O((m-1)(n/P)), where n is the total number of records in training data set D.

6 Conclusion

Many distributed algorithms have been proposed for classification algorithms like decision trees, but so far, there are few works in associative classification. In this paper, we have presented a distributed model which allows the class-association rules discovery in a shared-nothing architecture. Our solution embraces one of the fastest known sequential algorithms (FP-growth), and extends it to generate classification rules in a parallel setting. The divide-and-conquer nature of this latter facilitates the parallelization process, however, since the data set is distributed, global decision making becomes a difficult task. To avoid the replication of data in the sites, we have chosen to communicate the needed information. This exchange is made only in the first level of recursion, allowing each machine to subsequently process all its assigned tasks independently. At the end, a global classifier is built by all discovered rules, and applying a majority vote strategy. In order to evaluate this choices, it is imperative to carry out an experimental evaluation which permits us in the future to analyse several important costs: accuracy, scalability, memory usage, communication, synchronization, and also the load balancing.

References

1. Agrawal, R., Sharfer, J.: Parallel Mining of Association Rules. IEEE Transaction on Knowledge and Data Engineering 8(6), 962–969 (1996)
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rule. In: Proceedings of the 20th International Conference on Very Large Data Bases, pp. 487–499. Morgan Kaufmann, Santiago (1994)
3. Alipio, M.J., Paulo, J.A.: An experiment with association rules and classification: Post-bagging and conviction. In: Hoffmann, A., Motoda, H., Scheffer, T. (eds.) DS 2005. LNCS (LNAI), vol. 3735, pp. 137–149. Springer, Heidelberg (2005)
4. Buehrer, G., Parthasarathy, S., Tatikonda, S., Kurc, T., Saltz, J.: Toward terabyte pattern mining: An architecture-conscious solution. In: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 2–12 (2007)
5. Chawla, N., Eschrich, S., Hall, L.O.: Creating Ensembles of Classifiers. In: IEEE International Conference on Data Mining, pp. 580–581 (2001)

6. Chen, D., Lai, C., Hu, W., Chen, W.G., Zhang, Y., Zheng, W.: Tree partition based parallel frequent pattern mining on shared memory systems. IEEE Parallel and Distributed Processing Symposium (2006)
7. Cheung, W., Zaiane, O.R.: Incremental Mining of Frequent Patterns without Candidate Generation or Support Constraint. In: Seventh International Database Engineering and Applications Symposium (2003)
8. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1–12. ACM Press, Dallas (2000)
9. Javed, A., Khokhar, A.: Frequent Pattern Mining on Message Passing Multiprocessor Systems. Distributed and Parallel database, 321–334 (2004)
10. Li, W., Han, J.N., Pei, J.: CMAR: Accurate and efficient classification based on multiple-class association rule. In: Proceedings of the International Conference on Data Mining (ICDM 2001), San Jose, CA, pp. 369–376 (2001)
11. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the International Conference on Knowledge Discovery and Data Mining, pp. 80–86. AAAI Press, New York (1998)
12. Moonesinghe, H.D.K., Moon-Jung, C., Pang-Ning, T.: Fast Parallel Mining of Frequent Item-sets (Technical Report MSU-CSE-06-29). Dept. of Computer Science and Engineering, Michigan State University (2006)
13. Pramudiono, I., Kitsuregawa, M.: Shared nothing parallel execution of FP-growth. DBSJ Letters, v2 i1. 43-46 (2003)
14. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers, Inc., San Francisco (1993)
15. Thabtah, F.: Pruning techniques in associative classification: Survey and comparison. Journal of Digital Information Management 4, 202–205 (2006)
16. Thakur, G., Ramesh, C.J.: A Framework For Fast Classification Algorithms. International Journal Information Theories & Applications 15, 363–369 (2008)
17. Yu, K.M., Zhou, J., Hsiao, W.C.: Load balancing approach parallel algorithm for frequent pattern mining. PaCT, 623–631 (2007)
18. Zaiane, O., Lu, P.: Fast Parallel Association Rules Mining without Candidacy Generation. In: Proceeding of IEEE International Conference on Data Mining (ICDM 2001), pp. 665–668 (2001)
19. Zhou, J., Yu, K.M.: Tidset-based parallel FP-tree algorithm for the frequent pattern mining problem on PC clusters. In: Proceeding of 3rd international conference on grid and pervasive computing, pp. 18–28 (2008)

Distributed Multi-agent System for Solving Traveling Salesman Problem Using Ant Colony Optimization

Sorin Ilie[*] and Costin Bădică

Abstract. In this paper we present our approach and initial results for solving the Traveling Salesman Problem using Ant Colony Optimization on distributed multi-agent architectures. We introduce the framework including underlying architecture design, algorithms and experimental setup. Then we present initial scalability results that we obtained with the implementation of the framework using JADE multi-agent platform on a high-speed cluster network.

1 Introduction

Natural phenomena are inherently distributed, so we would expect distributed computing to have a lot of potential for the application of nature-inspired computing. In this context, we think that nature-inspired computing should allow a straightforward mapping onto existing distributed architectures, including multi-agent systems middleware. Therefore, to take advantage of the full potential of nature inspired computational approaches, we have setup the goal of investigating new distributed forms of Ant Colony Optimization (ACO hereafter) using state-of-the-art multi-agent technology.

We propose a multi-agent system architecture that allows the implementation of ACO in a parallel, asynchronous and decentralized environment. The architecture is applied for solving well-known Traveling Salesman Problem (TSP hereafter). The

Sorin Ilie · Costin Bădică
University of Craiova, Software Engineering Department,
Bvd.Decebal 107, Craiova, 200440, Romania
e-mail: {sorin.ilie,costin.badica}@software.ucv.ro

[*] This work was partially supported by the strategic grant POSDRU/88/1.5/S/50783, Project ID50783 (2009), co-financed by the European Social Fund Investing in People, within the Sectoral Operational Programme Human Resources Development 2007 2013.

novelty of our approach consists in: i) designing and implementing the computing system as a network of intelligent software agents ([7, 2]) that represents the problem environment, i.e. the nodes of the graph in the context of TSP and ii) reduction of ants management to messages exchanged between the agents.

Existing sequential computational approaches of ACO [4] are highly synchronous and require use of global knowledge. These create difficulties for designing distributed versions of ACO, thus hindering the potential gain of their implementation onto state-of-the art distributed multi-agent middleware.

For example, in [6] the authors propose a parallel, distributed, asynchronous and decentralized implementation of ACO. However, their approach requires maintenance of the globally best tour currently known using global update each time a better solution is found. Moreover, the authors' claim that "the parallelization and asynchronism has no effect on the accuracy, speed and reliability of the algorithm" is not supported by any experimental evidence.

The focus of this paper is to introduce ACODA and its functionalities including underlying architecture design, algorithms and experimental setup. Moreover, we present initial experimental results obtained with running ACODA for the TSP problem on a high-speed cluster network of 7 computers that was available for testing. The results are encouraging, as they clearly support the scalability of ACODA, thus confirming the feasibility of our proposal.

2 Background

ACO is inspired by behavior of real ants. When ants are searching for food, they secrete pheromone on their way back to the anthill. Other colony members sense the pheromone and become attracted by marked paths; the more pheromone is deposited on a path, the more attractive that path becomes. The pheromone is volatile so it disappears over time. Evaporation erases pheromone on longer paths as well as on paths that are not of interest anymore. However, shorter paths are more quickly refreshed, thus having the chance of being more frequently explored. Intuitively, ants will converge towards the most efficient path, as that path gets the strongest concentration of pheromone. Artificial ants are programmed to mimic the behavior of real ants while searching for food.

In this paper we propose a distributed approach to ACO called ACODA (Ant Colony Oprimization on a Distributed Architecture) and show how ACODA can be applied for solving TSP. The goal of TSP is to compute a shortest tour that visits each node of a complete weighted graph exactly once. The decision version of TSP is known to be NP-complete which basically means that it is very unlikely that a polynomial solution for solving TSP exists. So TSP is a very good candidate for the application of heuristic approaches, including ACO.

The main idea behind our ACODA approach is to provide a distributed architecture for modeling the problem environment. Artificial ants originating from the anthills that are located in the environment will travel to find optimal solutions, following ACO rules. In order to use ACO to solve TSP, the problem environment is conceptualized as a distributed set of interconnected graph nodes. Additionally, each graph node is also an anthill. Ants travel between nodes until they complete a tour. Once they return to their originating anthill, they mark the solution with pheromone by retracing their path.

Our model is based on the ACS algorithm presented in [4]. ACS is a sequential implementation of ACO in which it is preferred to move ants in parallel instead of moving each ant until it finishes its tour. Our approach presents a few differences due to the requirements of a truly distributed architecture based on asynchronous message passing and the avoidance of using global knowledge.

3 Mathematical Model

ACO rules determine the amount of pheromone deposited on edges, the edge chosen by each ant on its way, and how fast the pheromone deposited on each edge evaporates. For this purpose we use the mathematical model of ACO that is used in ACS.

In ACS, ant k located at node i decides to move to node j using "pseudo random proportional rule", following the equation:

$$j = \begin{cases} argmax_{l \in N_i}((\tau_{i,l})^{\alpha}(\eta_{i,l})^{\beta}), & \text{if } q \leq q_0 \\ J, & \text{otherwise} \end{cases} \tag{1}$$

where:

- α is a parameter to control the influence of $\tau_{i,j}$
- $\tau_{i,j}$ is the amount of pheromone deposited on edge (i,j)
- $\eta_{i,j}$ is the desirability of edge (i,j) computed as the inverse of the edge weight, i.e. $1/w_{i,j}$
- β is a parameter to control the influence of $\eta_{i,j}$
- q is a random variable uniformly distributed in $[0,1]$
- q_0 such that $0 \leq q_0 \leq 1)$ is a parameter
- J is a random node selected according to the probability distribution given by equation 2
- N_i represents the set of neighbors of node i

An ant located in node i will randomly choose to move to node j with the probability $p_{i,j}$ computed as follows:

$$p_{i,j} = \frac{(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}}{\Sigma_j(\tau_{i,j})^{\alpha}(\eta_{i,j})^{\beta}} \tag{2}$$

where:

- α is a parameter to control the influence of $\tau_{i,j}$
- β is a parameter to control the influence of $\eta_{i,j}$
- j represents a node reachable from node i that was not visited yet

Analyzing formula 1 it obviously follows that the ant makes the best possible move (as indicated by the learned pheromone trails and the heuristic information, i.e. the ant is exploiting the learned knowledge) with probability q_0, while it performs a biased exploration of the arcs with probability $(1-q_0)$. We chose $q_0 = 0$ because we concentrate on finding the best solution avoiding the convergence to a suboptimal one as much as possible. Therefore, in our approach only the probability function 2 shall be used to determine the next hop of an ant.

Better solutions need to be marked with more pheromone. So whenever an ant k determines a new tour V_k of cost L_k the ant will increase pheromone strength on each edge of the tour with a value that is inversely proportional to the cost of the tour.

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/L_k & \text{if edge } (i,j) \text{ belongs to found tour } V_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where L_k is the cost of the k-th ant's tour.

When an ant travels along a given path, this traveling takes an amount of time that is proportional with the travel distance (assuming the ants move with constant speed). As pheromone is volatile, if a real ant travels more, pheromone will have more time to evaporate, thus favoring better solutions to be discovered in the future. We can conclude that adding pheromone evaporation to our model can be useful, especially for solving a complex problem like TSP.

When an ant completes a tour it will retrace its steps marking the edges on the way with pheromone. The update will also take into account pheromone evaporation. Both evaporation and pheromone updates are implemented in ACS as follows:

$$\tau_{i,j} = (1-\rho)\tau_{i,j} + \rho\Delta\tau_{i,j}^{k} \tag{4}$$

where ρ is the evaporation rate $0 \leq \rho < 1$.

All ants use formula 1 to probabilistically determine their next step. Therefore they will often choose the edge with the highest pheromone, i.e. the exploration of less probable edges is low. The solution is to decrease the pheromone on edges chosen by ants, i.e. apply a local evaporation process. This has the effect of making them less desirable, increasing the exploration of the edges that have not been picked yet. Whenever an ant traverses an edge it applies local evaporation by updating pheromone as follows:

$$\tau_{i,j} = (1-\xi)\tau_{i,j} + \xi\tau_0 \tag{5}$$

where:

- ξ is the local evaporation rate $0 \leq \xi < 1$.
- τ_0 is the initial amount of pheromone on each edge

A good heuristics to initialize pheromone trails is to set them to a value slightly higher than the expected amount of pheromone deposited by the ants in one tour; a rough estimate of this value can be obtained by setting $\tau_0 = 1/(nC)$, where n is the number of nodes, and C is the tour cost generated by a reasonable tour approximation procedure [4]. For example we can set $C = nw_{avg}$ where w_{avg} is the average edge cost.

In order to observe the impact that evaporation has on the solutions, we have also considered a pheromone update scheme that does not include evaporation at all. In this case equations 5 and 4 are replaced by:

$$\tau_{i,j} = \tau_{i,j} + \Delta\tau_{i,j}^{k} \tag{6}$$

4 Architecture

In ACODA, the nodes of the graph are conceptualized and implemented as software agents [7]. For the purpose of this work, by software agent we understand a software entity that: (i) has its own thread of control and can decide autonomously if and when to perform a given action; (ii) communicates with other agents by asynchronous message passing. Each agent is referenced using its name, also known as agent id.

The activity carried out by a given agent is represented as a set of behaviors. A behavior is defined as a sequence of primitive actions. Behaviors are executed in parallel using interleaving of actions on the agent's thread with the help of a non-preemptive scheduler, internal to the agent [2].

Node design (see figure 1) must include a behavior for sending and receiving ants. Whenever an ant is received, the RECEIVE-ANT() behavior (see table 1) immediately prepares it and then sends it out to a neighbor node following ACO rules.

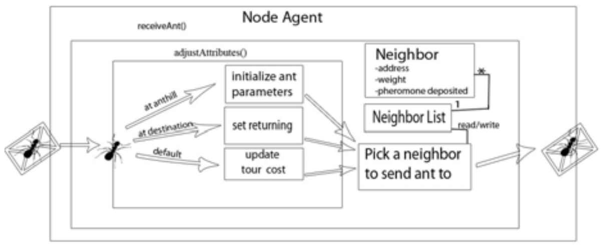


Fig. 1 Node structure in ACODA

Ants are represented as objects with a set of attributes: cost of the currently search path (which becomes L_k when the ant completes a tour), pheromone strength (value of $\Delta\tau_{i,j}^k$), returning flag, best tour cost (value of the currently best tour that the ant knows, based on its search history) and a list of node ids representing the path that the ant followed to reach its current location. The list is necessary for two reasons: i) the ant needs to retrace its steps in order to mark the tour with pheromone and ii) we need to avoid loops so only unvisited nodes are taken into account as possible next hops. Attributes are initialized when an ant is created and updated during the process of ant migration to reflect the current knowledge of the ant about the explored environment. For example, whenever an ant reaches a destination node (i.e. its anthill), the ant saves its currently best tour onto this node. So, whenever another ant from the colony travels through this node, it senses the environment and eventually updates its "knowledge" about the value of the currently best tour. So ants have the possibility to exchange through the environment (i.e. the set of graph nodes) not only pheromone information, but also information about the best solutions found so far.

Table 1 Algorithms for processing ant information

```
RECEIVE-ANT()
1. RECEIVE($ant$)
2. ADJUST-ATTRIBUTES($ant$)
3. SEND-TO($ant$,BEST-NEIGHBOR($ant$))

ADJUST-ATTRIBUTES($ant$)
1. if AT-ANTHILL($ant$) then
2.    if RETURNING($ant$) then
3.       INITIALIZE($ant$)
4.    else
5.          SET-RETURN-FLAG($ant$)
            ▷ calculate ant pheromone using equation 3.
6.          CALCULATE-ANT-PHEROMONE-STRENGTH($ant$)
7. UPDATE-BEST-TOUR()

BEST-NEIGHBOR($ant$)
1. if RETURNING($ant$) then
      ▷ update pheromone using equation 4 or 6.
2.    DEPOSIT-PHEROMONE()
3.    return LAST-VISITED-NODE($ant$)
   ▷ ant performs random choice according to equations 1 and 2.
4. $bestNeighbor \leftarrow$ RANDOM-CHOICE()
5. UPDATE-CURRENT-PATH-COST($bestNeighbor$)
   ▷ apply local evaporation of pheromone using equation 5.
6. LOCAL-EVAPORATE-PHEROMONE()
7. ADD-TO-VISITED-LIST($bestNeighbor$,$ant$)
8. return $bestNeighbor$
```

Nodes (represented as software agents) have the following parameters: value of ρ, value of ξ, best tour cost (the value of the currently best tour that the node knows, based on ants that traveled through this node) and a list of neighbor nodes. For each neighbor node we record the weight and the value of deposited pheromone of the corresponding edge. Parameters ρ and ξ are used to implement equations 4 and 5. Note that each node and each ant maintain their own values of the best tours they encountered so far. So, whenever an ant is traveling through a node, the ant and the node are able to exchange and update their best tour information accordingly.

In our approach each node creates its own ant population. Nodes calculate pheromone strength according to the tour cost (see equation 3) and also update the ants' pheromone strength attribute. Additionally, nodes set the returning flag for returning ants, exchange ant information with other nodes, deposit pheromone when needed, and update the cost of the current search path of an ant by adding the weight w of the edge that was chosen using equation 2.

The structure of a node is presented in figure 1. RECEIVE-ANT() behavior parses an ant message received by a node, adjusts ant's attributes using ADJUST-ATTRIBUTES() method and sends it out to the address determined by BEST-NEIGHBOR() method. This happens whenever the agent's message queue isn't empty [2]. ADJUST-ATTRIBUTES() method sets returning flag and calculates pheromone strength using equation (3) whenever an ant has completed a tour. Ants that have returned to the anthill are re-initialized.

BEST-NEIGHBOR() method uses equation (2) to determine the address of the node where to send the ant. When the ant returns to the anthill, this method sends the ant to the first node from its list of visited nodes, popping it from the list, and deposits the ant's pheromone. This method also implements evaporation using equations 5 and 4.

5 Experiments

In order to facilitate experimentation with ACODA, we created a distributed platform based on JADE framework [2] than can be configured to run on a computer network. The focus of the experiments was to analyze scalability of ACODA. For that purpose we used the platform to run the ACODA version of the ACS algorithm on a high-speed cluster network, by varying the number of available machines from 1 to 7. Note that ACODA can be configured to run distributed versions of other ACO algorithms ([4]).

5.1 Setup

An experiment is structured as a fixed number of independent experimental rounds. A round consists of one execution of ACODA for a given initial set of parameters. All parameters are initialized at the beginning of each round. Experimental data are collected during each round. At the end of the experiment (i.e. when the fixed number of rounds was reached) these data are post-processed to calculate experimental performance measures that are needed to appreciate scalability of ACODA.

ACODA is a distributed platform. Setting-up and running ACODA on several computers assumes two stages: (i) initialization (or bootstrap) stage; and (ii) execution stage.

During the initialization stage: JADE platform is started, a number of containers are created (typically one container for each available machine), and finally node agents are created and evenly distributed on available containers of the JADE platform.

The experiment is ran during the execution stage. Execution of an experiment is controlled using special control messages. Control messages are distinguished from ant exchange messages using their conversation id. Command messages have their conversation id set to "command", while ant exchange messages have their conversation id set to "ant". Command messages are given higher priority than ant exchanging messages.

An experimental round starts when all node agents receive a "start" command, and ends when all node agents receive a "stop" command. A designated node agent – called *MasterNode* is responsible with controlling the experimental rounds (i.e. issuing of "start" and "stop" commands) and calculating performance measures. The *MasterNode* counts the total number of ants that it receives. When this number reaches a given maximum value M, *MasterNode* stops the current experimental round (issuing a "stop" command) and starts a new fresh experimental round (issuing a "start" command).

Duration T_i of an experimental round i is recorded by *MasterNode* agent as time elasped between issuing a "start" and "stop" command. While the distributed ACO algorithm is running, minimum tour costs are passed from node to node via ants. Whenever a node agent updates its current value of the minimum tour, the time elapsed since the node received the "start" command is also recorded. When an experimental round is finished, *MasterNode* agent centralizes the experimental data collected during the round. Note that this centralization is done only for experimental purposes and does not affect the decentralized nature of the ACODA architecture.

Note that we cannot assume that the best tour recorded by the *MasterNode* is actually the best tour computed during an experimental round. So we need to determine the minimum of the best tours computed by all node agents together with the time at which this tour was discovered.

Let us suppose that we have n node agents and k experimental rounds. For each node agent $j \in \{1,2,\ldots,n\}$, let $t_{i,j}$ be the time of the last update of the best tour performed by node agent j in round $i \in \{1,2,\ldots,k\}$, and let $v_{i,j}$ be the cost of the corresponding tour. *MasterNode* agent will collect values $v_{i,j}$ and $t_{i,j}$ and will determine the solution v_i and associated time t_i in round i as shown in first two rows of table 2.

The experimental data that were acquired during all the rounds of an experiment are post-processed to calculate performance measures as shown in last five rows of table 2.

Table 2 Calculation of performance measures

$v_i = \min_{j=1}^{n} v_{i,j}$	v_i is the solution found by round i.
$t_i = \min_{j=1}^{n} \{t_{i,j} \| v_{i,j} = v_i\}$	t_i is the time in which solution was found by round i.
$v_{avg} = \frac{\sum_{i=1}^{k} v_i}{k}$	v_{avg} is the average value of solutions found in all rounds.
$t_{avg} = \frac{\sum_{i=1}^{k} t_i}{k}$	t_{avg} is the average time in which solutions were found in all rounds.
$v_{min} = \min_{i=1}^{k} v_i$	v_{min} is the best solution found after carrying out all rounds.
$t_{min} = \min_{i=1}^{k} \{t_i \| v_i = v_{min}\}$	t_{min} is the minimum time in which the best solution was found in all rounds; it indicates when the best solution was first found.
$T_{avg} = \frac{\sum_{i=1}^{k} T_i}{k}$	T_{avg} is the average execution time of the rounds in an experiment.

5.2 Initial Performance Analysis

We ran several experiments using the following benchmark TSP maps selected from TSPLIB [5]: eil51, st70, kroA100, ch150. Note that the number in the map name indicates the number of nodes of the map. These maps were chosen to experiment with different values of w_{avg} and $\Delta w = w_{max} - w_{min}$ where Δw is the difference between the maximum and the minimum edge weight and w_{avg} is the average edge weight.

The ACO parameters were set to the values recommended by [4] for the ACS algorithm on which our approach is based. The total number of ants is equal to the number of nodes n (i.e. for each node the ant population consists of a single ant), $\tau_0 = 1/(n^2 w_{avg})$, $\rho = \xi = 0.1$, $\alpha = 1$, $\beta = 5$.

We define an ant move as the action of transferring an ant from a source node to destination node along a given edge. The number of ant moves was chosen according to the proposal from [4]. There, 1000 moves per ant were chosen for a 19 node map. Taking into account the sizes of our maps and scaling up proportionally, we determine a number $M = 10000$ of ant moves for our experiments.

We ran 10 rounds for each experiment on networks of 1, 4, and 7 computers with dual core processors at 2.5 GHz and 1GB of RAM memory. These workstations were interconnected using a high-speed Myrinet interconnection network at 2Gb/s.

Table 3 Experimental results. The number in the name of the map represents the number of nodes and the number written between brackets represents the optimum tour cost.

		1 comp					4 comp					7 comp				
map	e	v_{avg}	$t_{avg}[s]$	v_{min}	$t_{min}[s]$	$T_{avg}[s]$	v_{avg}	$t_{avg}[s]$	v_{min}	$t_{min}[s]$	$T_{avg}[s]$	v_{avg}	$t_{avg}[s]$	v_{min}	$t_{min}[s]$	$T_{avg}[s]$
eil51	1	487	40.5	483	58.4	83.4	479	12.8	473	0.9	57.3	482	15.8	471	2	38.1
(426)	0	459	33.9	452	27.9	79.3	462	46.7	449	64.3	60.6	449	12.8	442	10.9	39
st70	1	729	67.3	712	38.2	127.1	720	32.4	701	45.2	90.3	688	25.3	684	6.4	56.2
(675)	0	685	45.9	676	66	127.4	727	43.8	716	35.6	91.3	685	28.1	679	20.8	54.6
kroA100	1	23319	49.9	22036	35.3	236.5	24427	13.6	24207	4.8	142	22602	53.8	22286	31.6	92.3
(21282)	0	22590	38.7	22152	31.5	274.7	22678	22.8	22232	15.6	145.4	22997	29.1	22605	77.3	92
ch150	1	8069	25.7	7992	524.8	572.7	8041	139	7907	166.9	300.5	7729	17.4	7638	26	194.2
(6528)	0	6810	33.7	6753	244.6	579.8	7651	135	7434	301.6	291.3	6754	71.7	6692	57.5	182.8

Our experiments (see table 3) clearly show that increasing the number of computers determines the decrease of the overall execution time of the algorithm, thus supporting our claim that ACO can be implemented in a parallel, asynchronous, decentralized and distributed way without significant increase of the message overhead when the number of machines scales up. This does not come in contradiction with the claims in [4] that a fine grained parallelization is not feasible because the authors use ants as pheromone updating functions, while we use the node agents to accomplish this task. In their view each ant holds a copy of the map and uses it to mark its tour. Then the maps are compared and synchronized, which causes a large communication overhead that surpasses any computation time gained through parallelization. So even if a fine grained parallelization in the sense of one thread per ant is not feasible, we proved in this paper that a fine grained parallelization in the sense of one thread per node is feasible. Note that we do not claim this approach has very little communication and thread commutation overhead as this in not the case. We do, however, claim the overhead does not increase enough to overwhelm the execution time gained by adding more computers, and this claim is supported by our initial experimental results.

In our experiments we considered both evaporation scheme suggested by [4] for ACS (equation 4; see rows marked with $e = 1$ on table 3) and absence of evaporation (equation 6; see rows marked with $e = 0$ on table 3). Note that the algorithm has no significant benefit from using the ACS evaporation scheme with the recommended parameter values. As a matter of fact in most cases evaporation decreases the quality of the solutions. However, further study is needed in order to establish wether the evaporation parameters should be adjusted or a completely new approach should be developed.

6 Related Work

TSP is a classic benchmark problem for heuristic search algorithms. With the advent of distributed computing technologies, distributed versions of heuristic algorithms for TSP were also proposed. Based on our literature review, there are very few works that propose ACO-based distributed TSP algorithms. Moreover, there are even fewer proposals that utilize recent advances of multi-agent systems middleware for ACO-based TSP ([6]). Nevertheless, we could find references to multi-agent approaches to ACO algorithms for other combinatorial optimization problems ([8], [1], [3]).

A closely related approach to agent-based distributed ACO is presented in [6]. There, both graph nodes and ants are implemented as JADE software agents. Ants centralize information about pheromone deposits and nodes' best tour cost through a single ACL message exchange per node [2]. This procedure adds up to $2n$ messages per tour per ant, where n is the number of nodes. Each ant has to notify the node about its next hop and the cost of its tour in order for the node to be able to update its pheromone levels. This generates other n messages. When an ant completes a tour, it compares the tour cost with the collected best tours from the nodes. A best tour synchronization is triggered for all the nodes if a better tour has been found. This brings an additional overhead of n messages. So, [6] approach requires at most $4n$ messages per tour per ant, while our approach requires at most $2n$ messages: n messages (ant moves) to complete a tour and n messages to deposit the pheromone. We avoid the additional overhead of sending to nodes all the information necessary to carry out their tasks, as in ACODA this information is already contained in ant messages exchanged between nodes.

ACODA implementation reported here is based on the sequential ACS algorithm presented in [4]. There are however three notable differences: i) We avoid iterations. An iteration lasts until every ant has found a tour. It would be time consuming to synchronize all nodes, so it would cancel the main benefits of an asynchronous, distributed architecture; ii) In ACS, best tours are compared after each iteration, thus allowing only the best ant to mark its tour. Again, this would require synchronization and centralization so we avoid it by allowing all ants to mark their tours; iii) In ACS ants move synchronously, taking one step at a time, while in our approach ants move asynchronously.

Papers [1] and [3] propose JABAT – a JADE-based middleware for agent teams. JABAT supports distributed implementation and collaboration of population-based optimization algorithms. In particular, JABAT was applied to TSP. There is however a very important difference between JABAT and ACODA. JABAT agents represent improvement algorithms, which basically means that improvement algorithms are sequential and they cooperate for solving a problem in a distributed way. ACODA agents provide a natural distributed model of the problem environment that is suitable for distributed ACO algorithms. Moreover, we could not find scalability studies referring to JABAT.

Paper [8] proposes a JADE-based multi-agent environment for dynamic manufacturing scheduling that combines intelligent techniques of ACO and multi-agent coordination. However, the focus in [8] is to evaluate the impact of ACO intelligence on multi-agent coordination, rather than to utilize multi-agent middleware for improving ACO. Therefore ACO intelligence is embedded into job and machine agents that have the role of ants, which is different from ACODA where ants are passive objects exchanged by software agents providing a distributed model of the problem environment.

7 Conclusions

In this paper we introduced a new multi-agent framework for distributed ACO. An initial implementation of the framework using JADE multi-agent platform was outlined. This implementation followed the ACO approach initially proposed for the ACS system. The implementation was evaluated on sample benchmark TSP problems. Experimental results were encouraging, as they clearly support the scalability of our system, thus confirming the feasibility of our framework. There are few directions that we would like to follow as future works: i) strengthen the scalability results by experimenting with larger TSP problems on larger computer networks; ii) support the generality of our framework by considering other forms of ACO rather than only ACS; iii) investigate new forms of distributed ACO.

References

1. Barbucha, D., Czarnowski, I., Jedrzejowicz, P., Ratajczak, E., Wierzbowska, I.: Jade-based a-team as a tool for implementing population-based algorithms. In: Proc. 6th International Conference on Intelligent Systems Design and Applications: ISDA 2006, pp. 144–149. IEEE Computer Society, Los Alamitos (2006)
2. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons Ltd., Chichester (2007)
3. Czarnowski, I., Jedrzejowicz, P., Wierzbowska, I.: A-team middleware on a cluster. In: Håkansson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2009. LNCS, vol. 5559, pp. 764–772. Springer, Heidelberg (2009)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)
5. Reinelt, G.: Tsplib - a traveling salesman library. ORSA Journal on Computing 1, 376–384 (1991), http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
6. Ridge, E., Curry, E., Kudenko, D., Kazakov, D.: Parallel, asynchronous and decentralised ant colony system. In: Proc. 1st International Symposium on Nature-Inspired Systems for Parallel, Asynchronous and Decentralised Environments, NISPADE 2006 (2006)
7. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley & Sons Ltd., Chichester (2002)
8. Xiang, W., Lee, H.P.: Ant colony intelligence in multi-agent dynamic manufacturing scheduling. Engineering Applications of Artificial Intelligence 21(1), 73–85 (2008)

Optimal Message Interchange in a Self-organizing Multi-agent System

Antonio González-Pardo, Pablo Varona, David Camacho, and
Francisco de Borja Rodriguez Ortiz

Abstract. Over the last decade there has been a growing interest on Intelligent Agents and Multi-Agent Systems (MAS) in several fields such as Artificial Intelligence (AI), Software Engineering, Psychology, etc... Different problems can be solved in these fields by creating societies of agents that communicate with each other. Nevertheless, when the number of agents is large and the connectivity is extensive, the system suffers from overhead in the communication among agents due to the large number messages exchanged. This work addresses the search for an optimal communication topology to avoid these situations. This optimal topology is characterized by the use of a redirecting probability in the communication. The redirection of a communication is performed before the execution of the MAS. Once agents start the execution, the topology is fixed and remains unchanged. This characteristic is useful in those systems where a given topology can not be changed as, for example, in wired networks. On the other hand, in the proposed solution agents contain a local message discrimination process as a function of the sender of the message. Experiments show an important improvement in terms of a reduction in the number of iterations needed to solve the problem and also in the number of messages exchanged.

1 Introduction

A problem using Multi-Agent Systems appears when the system contains a large number of agents connected extensively. Agents need to stablish connections to interchange messages and realize a set of goals, or tasks, for which the system is

Antonio González-Pardo · Pablo Varona · David Camacho
· Francisco de Borja Rodriguez Ortiz
Departamento de Ingeniería Informática, Escuela Politécnica Superior,
Universidad Autónoma de Madrid
C/Francisco Tomás y Valiente 11, 28049 Madrid, Spain
e-mail: {antonio.gonzalez,pablo.varona,david.camacho,
f.rodriguez}@uam.es

designed. Depending on the topology of the network created and the number of agents in the system, overhead could appear due to the large number of messages interchanged, for example in broadcast topologies. To deal with the communication overhead, this paper studies the influence of the topology of the communication among agents. The main characteristic of the agents in the proposed solution is the discrimination of messages as a function of the sender in combination with an optimal topology that provides a faster propagation of the information without overhead the system.

The overhead problem in Multi-Agent Systems has been studied before and it is independent on the architecture used, [Jurasovic et al., 2006]. For example, in mobile agents it has been proposed the use of a dynamic agent distribution that reduces the communication between servers, [Jang and Agha, 2006]. This distribution is based on the idea of allocating agents with high communication rate between them, in the same server. This is useful because inter-node communication does not affect the overhead. But with this approach, a new problem appears which is how to select the agents with heavy traffic between them to migrate all of them into the same server. To solve this problem a new algorithm is proposed in [Miyata and Ishida, 2008].

Other works have addressed the discrimination of messages, [Sugawara and Lesser, 1993] and [Sugawara and Kurihara, 1998]. In these works agents learn what messages have high probability of being important based on rules created by each agent. Analysing the history of past inferences, agents create local rules that allow them to behave in a different way depending on the received message.

The discrimination technique described in this paper is a bio-inspired method based on the bursting activity of living neurons, [Szücs et al., 2003]. The bio-inspiration comes from the role of identity codes in neural systems which has been extensively with realistic models in [Latorre et al., 2006].

The main contribution of this paper is the search of an initial communication topology for a self-organizing Multi-Agent System. Static agents discriminate messages depending on the sender but their behaviour is not influenced by inference rules or learning procedures. This discrimination procedure has been applied previously in a new self-organizing neural network paradigm [Latorre et al., 2010] where the communication topology is dynamic and changes during the execution of the system. Although there are other approaches based on mobile agents ([Jang and Agha, 2006]), this work uses the optimization of the network topology to reduce the communication impact in the network. The communication topology suggested in this work deals with the communication overhead problem. This topology is based on a regular topology where each edge is redirected randomly according to a specific probability. Once the topology is created, it does not change during the execution of the system. Finally, our approach is illustrated by solving a jigsaw puzzle problem.

2 Description of the Model

This section describes the agent model, and the topology used to allow the optimal communication in the system.

2.1 Description of the Agent Model

Agents modelled in the system are static lightweight agents with a very low autonomy. However, most of the approaches of swarm and collective intelligence agents have the same behaviour and agents are indistinguishable. This work does not use this idea and each agent contains an unique identification that enables the differentiation between them. This characteristic is a key concept of the system because it will allow the discrimination of information as a function of the sender of the message.

In Multi-Agent Systems, agents have an incomplete information for solving the problem at hand. The piece of information contained in each agent is called, in this work, *Agent Information*, *[AIn]*, and the identification of each agent is called *Agent Identification*, *[AId]*. Both concepts compose the message that will be sent to the agents neighbourhood. Therefore, messages have the following structure $[AId|AIn]$. The first part of the message contains information that allows the identification of the sender, and the rest are data needed to solve the problem. Note that information stored in each part depends on the problem, *[AIn]* could contain much more information than *[AId]*. In these situations where Agent Information is large, it is very important the discrimination based on the recognition of the sender to avoid analyzing non relevant information.

The tasks performed by the agents will depend on the content of the messages received. That is, the behaviour of the agents will depend on who sends the information because depending on the sender, the message will be ignored or not. Nevertheless, the set of actions performed by the agents is always the same, there are not self-adaptation nor learning in its behaviour. Figure 1 shows the behaviour of agents.

Agents contain a temporal memory, called *local informational context*. Using this memory, agents can retain, during a certain period of time, a set of messages received by their neighbourhood. All received messages are stored in this memory, which means that it is not important whether the sender is recognized or not, because the message will be added to that memory in both cases. The fact that an agent does not recognize the sender of a message does not mean that any agent belonging to its neighbour will not recognize the sender. In the case where the received messages exceed the memory size, the oldest messages are replaced by the most recently ones. Finally, the information of the agent is added to this memory and all the content of the memory is sent.

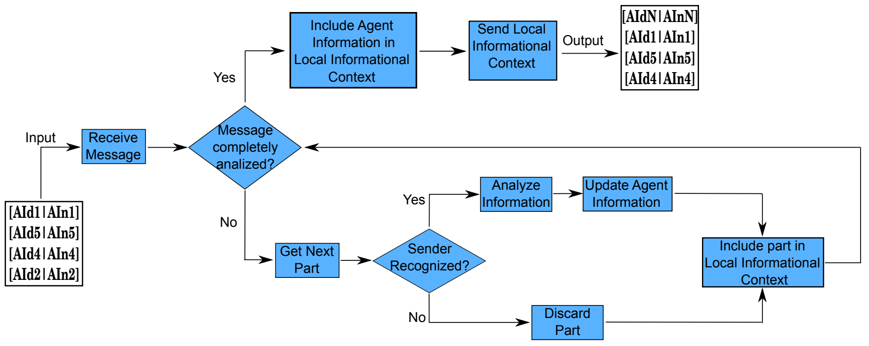


Fig. 1 Flow diagram that describes the behaviour of agent N. Each agent receives a set of messages, and starts analysing the sender of each message ($[AId]$). If the sender is not recognized, the message is discarded without analyzing the information contained ($[AIn]$). This discrimination procedure is really useful in environments where $[AIn]$ is bigger than $[AId]$ because agents does not analyze uninteresting $[AId]$ saving processing time.

2.2 *Network Topology*

The organization of any MAS can be analyzed from different perspectives which goes from hierarchical structures to completely random structures. However, it is necessary to provide some kind of structure to allow the interactions between agents. This organization can be studied from the point of view of a network topology.

In this work the topology selected to connect different agents is the Ring topology. This topology depends on a parameter named *connectivity degree (k)* that defines the number of connections that each node will have. Figure 2.a shows an illustrative example of a Ring Topology with $k = 1$, and 9 agents (nodes).

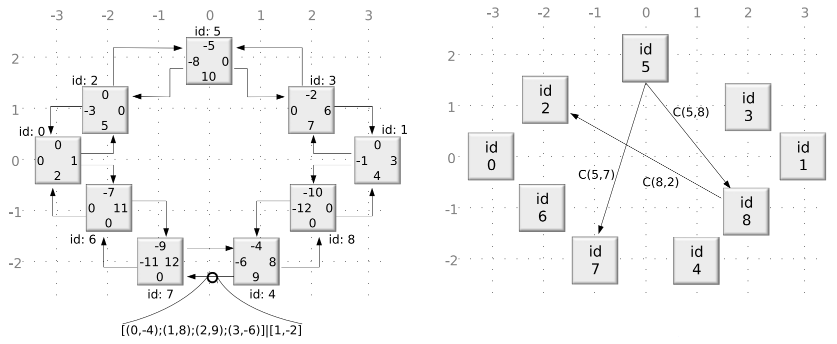


Fig. 2 Details of the communication topology. Figure on the left shows an example of a ring topology with 9 agents, and connectivity degree (k) 1. The figure on the right is a graphical representation of communication costs in the topology.

As it can be seen in Figure 2.a, given a particular connectivity degree, k, each node will be connected to $2*k$ agents. Note also that the connections are unidirectional. This means that if node A is the origin of a connection to node B, node B is not able to use that connection to send a message to node A, and thus node B need another connection that goes from B to node A.

One of the problems of regular topologies is that the propagation of the messages in the network is very slow. This is produced due to the characteristic path length is very high. This metrics represent the mean length of the paths in the network.

For a specific value of k, each node will be connected to its $2*k$ nearest nodes. Increasing the k of the topology, nodes will be highly clustered and the characteristic path will be decreased because the network will contain more edges. The minimum value for k is 1, in this situation each node will be connected to its 2 nearest nodes. On the other hand, the maximum connectivity corresponds with the broadcast mode, and this situation is produced when the connectivity degree is $n/2$, where n is the number of nodes in the network.

It is important to take into account that by increasing the connectivity degree of the topology, the system will converge in less iterations but the number of messages exchanged will also increase. Therefore, the value chosen for k is a trade off between the number of messages exchanged and the number of iterations to converge.

2.3 Searching an Optimal Communication Topology

As it was described in Section 2.2, the main problem in regular topologies is the value of the characteristic path length. From a communicational point of view, the lower the value of the characteristic path length is, the faster the messages are propagated.

In order to reduce the characteristic path length, a new parameter p is introduced in the system. This parameter is the probability of redirecting a connection. Given a specific value of p, each connection is analyzed and connections will be redirected with a probability p.

Using this new parameter, the connections are redirected to a randomly chosen node in the network, so the characteristic path length is reduced. Moreover, considering only the connectivity degree of the node (k, see Section 2.2) there is only one topology for each connectivity degree. Considering the parameter p, for each k and p fixed there are a family of topologies.

The probability of redirecting a connection was introduced by [Watts and Strogatz, 1998] in the definition of a Small World Network. It is important to note that for values of p close to 0 the network will have a regular topology, because few connections will be redirected. On the other hand, values close to 1 will generate Random Networks.

Small World Networks are characterized by having short path lengths between their nodes, which provides fast propagation of information, and high clustering rate that makes the network robust to attacks. This type of topology has been used to optimize the mutual information exchanged between nodes [Dominguez et al., 2009].

Finally when all connections have been treated by a fixed probability, agents start the execution and the topology created does not change.

3 Solving Puzzles Using an Optimal Communication Topology

There are many practical applications in which some agents need to know certain information about their environment in order to solve the problem. These applications can take advantage of the system described in this paper because the aim is to determine the optimal communication topology to solve the system by reducing the number of iterations and avoiding the overhead problem. Some examples of those applications, where our approach could be used, are scheduling problems, job shop problems, routing problems, etc.

The application selected to apply our approach tries to solve a puzzle where each piece needs to know which pieces match with its sides. Note that the neighbour of a piece does not necessarily contain pieces that matches with it. Using the technique described in this paper, each piece sends its information to the minimum number of agents to solve the puzzle in a reasonable amount of time.

Using the model to solve a puzzle, each agent will represent a single piece of the puzzle. Puzzles compose an image, and each piece contains a part of that global image. For that reason the agent information will be that part of the whole image contained in the piece.

3.1 Piece-Agent Codification

In order to model the shape of a side, each shape is represented by an integer value. For example, the border of the puzzle is represented as 0. With this approach, two pieces are compatible if the addition of the corresponding side values are equals 0 and none of these values are 0. This means that a piece with a side value X will match with the piece with value $-X$.

Using this representation, a message is relevant for an agent when the message contains, at least, one agent information compatible with any receiver agent sides. When a relevant message is received, the agent extracts the information about the compatible agent and memorizes that it must be connected to the sender through the corresponding side.

Each agent will be referred by its coordinates in a plane. That identification is unique because the agents are static and two agents cannot be located in the same place.

When an agent processes a relevant message, it updates its agent information, as shown in Figure 1. This means that agents need to delete from its agent information the side just matched. Each piece has four sides: upper side, right side, bottom side and left side. Those sides are identified by 0, 1, 2 and 3 respectively. In Figure 2.a, the piece with *AId* 4, which is located in $(1,-2)$, sends the message *[(0,-4);(1,8);(2,9);(3,-6)]*|*[1,-2]* to its neighbour.

3.2 Piece-Agent Communication

In real world, the communication between two partners has a cost which can be expressed in terms of monetary, time or performance cost. In this work, the cost of sending a message is proportional to the distance between the sender and the receiver. The cost of a communication is defined by equation 1.

$$C(i,j) = \lfloor \frac{EuclideanDistance(i,j)}{\min(EuclideanDistance(l,m))} \rfloor \forall l,m \mid l \neq m \qquad (1)$$

The cost of a communication between two agents is the first positive number that exceeds the euclidean distance between both agents, normalized by the minimum distance in the system. Costs resulting from Equation 1, describe the number of iterations that a message will take to go from the sender to the receiver. The minimum cost is 1 and means, that receiver will have the message available in the next iteration. Figure 2.b shows a representation of the cost in the network.

4 Experimental Results

This section describes the different experiments carried out in this work. As it was described in section 3, the model has been adapted to solve a puzzle as an illustrative example.

In order to analyse the impact of the probability on the performance of the system, the original topology will be tested with with 21 different values of this probability. The values go from 0 to 1 with increases of 0.05.

All the experiments carried out in this work are based in a puzzle with 100 pieces, and all the shapes of the pieces are unique. This means that each piece will match, exactly, with one piece for each side. The algorithm initializes the agents and creates the topology defined by a fixed probability. Finally, agents are executed to solve the puzzle.

Figure 3 shows the iterations taken by the system to solve the puzzle. Those charts represents the number of iterations taken by the system to solve the problem for a specific probability value. The value of k is 3 for Figure 4.A, while the rest of figures corresponds to executions with $k = 8$.

In order to analyze how the memory size affects the performance of the system in terms of iterations taken to solve the puzzle, four experiments have been carried out. All experiments try to solve a puzzle with 100 pieces, and all pieces have $k = 3$. The memory size changes in each experiment, and it takes values 3, 8, 15 and 30. Figure 4 shows the performance of these experiments. From this figure, it is deduced that a larger memory provides an important improvement in the system with lower values of probability.

Furthermore, there is a limit probability from which there is no important improvement in the system. This limit is located between 0.3 and 0.4 and means that with this probability the characteristic path is very low.

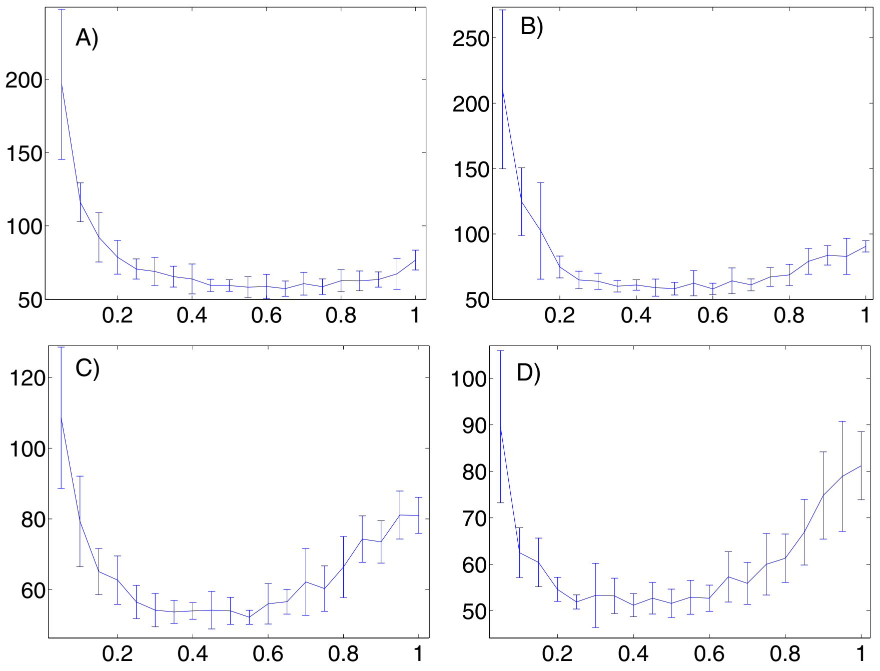


Fig. 3 Iterations per probability resulting from the execution of the system with different values for the memory and the connectivity degree of the topology. The system is composed by 100 agents that represent a puzzle with 100 pieces. For a specific probability, the system has been executed 10 times. Y-axis represents the number of iterations taken by the program, while X-axis shows the value of probability p. Figure A shows the performance of the system with memory 3 and $k = 6$. Figure B represents the system with memory 8 and $k = 3$. Finally, performance of the system with $k = 3$ and memory 15 and 30 is shown in Figures C and D.

Finally, as it was stated previously, the memory size affects the performance of the system. Nevertheless, the performance of a system with size memory 15 is very similar to the performance with memory 30. This fact suggest the existence of a limit in the memory size. This limit must be studied in future works because it is important to build a system that optimizes resources.

In order to measure the overhead problem in the system, the number of messages sent by the agents is compared with the a broadcast situation. The broadcast mode guarantees a solution, because each agent will received a message from the rest but the cost of sending a message affects the number of iterations taken to achieve the solution. Apart from this, designing a broadcast mode in a real system, for example, communication between hosts, it is not useful because to build a system with these characteristic could be very expensive.

Figure 5 shows the performance of the system taken into account the total number of messages sent. Although these messages could have different length, (because if

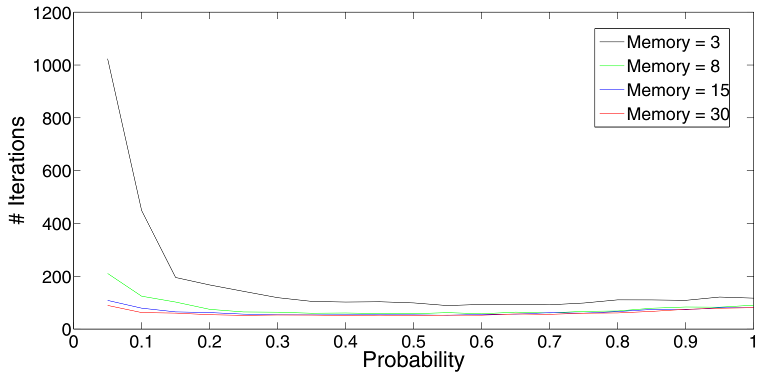


Fig. 4 Performance of the system, in terms of number of iterations taken to solve the puzzle, with memory 3, 8, 15 and 30. All executions try to solve a puzzle with 100 pieces, and each agent has $k = 3$.

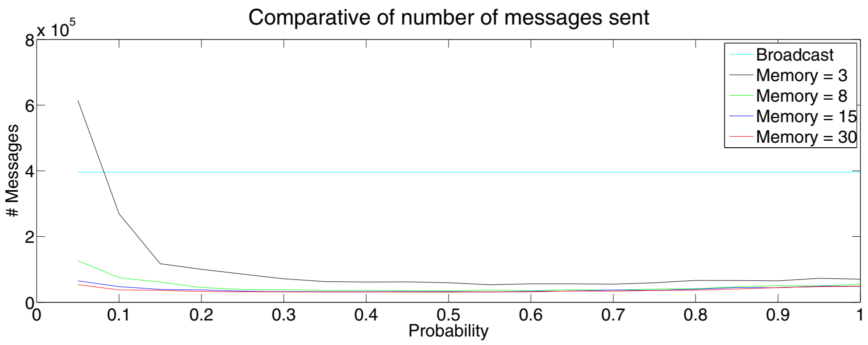


Fig. 5 Difference between the system and the broadcast. The connectivity degree of the topology in the broadcast mode is 50 ($n/2$), in the rest of executions the value of k is 3.

a piece has a side matched, the information belonging to that side is not sent) in these experiments, the suggestion that all messages have the same length is taken into account. This is not strange because, in this problem, the difference between the largest message and the smaller message is not relevant. Nevertheless, a more specific study on the total number of messages is required.

From the four experiments described in this section, the only execution that exceeds the number of messages sent by the broadcast mode is with memory 3 and lower probability (0.05). This result is expected because in this situation, the topology is very similar to the regular ring (with high characteristic paths) and the message size is very small. Nevertheless, with a little change in the probability (from 0.05 to 0.1) the number of sent messages is reduced around 56%. The rest of executions, independently of the probability, improve the number of messages sent in the broadcast situation.

5 Conclusions

This paper studies the problem of communication overhead in a network of agents analyzing the topology of the communication. The characteristics of this topology try to meet the following goals. On one hand, to reduce the geodesic path of the network in order to propagate messages in fewer iterations. On the other hand, to reduce the number of messages exchanged among agents and, in that way, to avoid the overhead problem.

This work modifies the regular topology with a specific probability, as a Small World Network. Nevertheless, it is important to notice that this redirection of communication channels is only performed once and when agents start the execution the topology is fixed. This concept is similar to a computer network where, once there is a connection between two devices or sub-networks, the connection is not redirected because the wires are physically inaccessible or the cost of redirecting is expensive.

The results show that modifying the probability of redirection, there are important improvements in the number of iterations needed to solve the problem (as compared to the regular topology). The value of the optimal probability depends on the connectivity degree, Figure 3 shows that for $k = 3$ the optimal probability is around 0.25 and 0.45, and for $k = 6$ the optimal probability is located between 0.55 and 0.65. As there is no direct relation between the connectivity degree and the probability, a deeper study is needed.

The memory of the agents plays an important role in the performance. Figure 4 shows that with larger values for the memory, the system solves the problem in fewer iterations. Taking into account the number of messages sent in the system, Figure 5 shows that with minor modifications in the communications (low probability), the number of messages sent is less than the number of messages sent in the broadcast situation.

Finally, the experimental results show how our approach allows to reduce significantly the communication overhead in a MAS by modifying the regular topology of the agent communication network.

Acknowledgements. This work has been partially supported by the Spanish Ministry of Science and Innovation under TIN 2007-65989, TIN 2007-64718 and MICINN BFU2009-08473.

References

[Dominguez et al., 2009] Dominguez, D., Gonzalez, M., Serrano, E., Rodriguez, F.B.: Structured information in small-world neural networks. Physical Review E (Statistical, Nonlinear, and Soft Matter Physics) 79(2), 021909 (2009)

[Jang and Agha, 2006] Jang, M.-W., Agha, G.: Agent framework services to reduce agent communication overhead in large-scale agent-based simulations. Simulation Modelling Practice and Theory 14(6), 679–694 (2006)

[Jurasovic et al., 2006] Jurasovic, K., Jezic, G., Kusek, M.: A performance analysis of multi-agent systems. ITSSA 1(4), 335–342 (2006)

[Latorre et al., 2006] Latorre, R., Rodríguez, F.B., Varona, P.: Neural signatures: multiple coding in spiking-bursting cells. Biological Cybernetics 95, 169–183 (2006)

[Latorre et al., 2010] Latorre, R., Rodríguez, F.B., Varona, P.: Signature neural networks: Definition and application to jigsaw puzzle solving. IEEE Transaction on Neural Networks (2010) (page to appear)

[Miyata and Ishida, 2008] Miyata, N., Ishida, T.: Community-based load balancing for massively multi-agent systems. In: Jamali, N., Scerri, P., Sugawara, T. (eds.) MMAS 2006, LSMAS 2006, and CCMMS 2007. LNCS (LNAI), vol. 5043, pp. 28–42. Springer, Heidelberg (2008)

[Sugawara and Kurihara, 1998] Sugawara, T., Kurihara, S.: Learning message-related coordination control in multiagent systems. In: Selected Papers from the 4th Australian Workshop on Distributed Artificial Intelligence, Multi-Agent Systems, pp. 29–44. Springer, Heidelberg (1998)

[Sugawara and Lesser, 1993] Sugawara, T., Lesser, V.: Learning coordination plans in distributed problem-solving environments. Technical report. In: Twelfth International Workshop on Distributed Artificial Intelligence (1993)

[Szücs et al., 2003] Szücs, A., Pinto, R., Rabinovich, M., Abarbanel, H., Selverston, A.: Synaptic modulation of the interspike interval signatures of bursting pyloric neurons. Journal of neurophysiology 89(3), 1363–1377 (2003)

[Watts and Strogatz, 1998] Watts, D.J., Strogatz, S.H.: Collective dynamics of ’small-world’ networks. Nature 393(6684), 440–442 (1998)

Part V
Pervasive, Ambiental and Grid Computing

Compromised Resources Tolerance in Grid Computing Systems

Ahmed Bendahmane, Mohammad Essaaidi,
Ahmed El Moussaoui, and Ali Younes

Abstract. In the past ten years, grid computing has been heavily researched and developed. At the present, grid technology has matured enough to enable the realization of workable and commercial infrastructures, platforms and tools, ready to be used in production environments ranging from scientific to commercial application domains. As grid resources are used outside of organizational boundaries, it becomes increasingly difficult to guarantee that a resource being used is not malicious in some way. However, the critical grid application domains require many protections against malicious entities or massive attacks. It is then necessary to detect and tolerate malicious behavior of grid resources (or resources provider) in order to enhance the efficiency of grid security architecture. In this paper, we will propose and discuss an attack-tolerance mechanism based on sabotage tolerance technique. The grid broker service will build the reputation of all resources provider, by observing their trustworthiness from their previous results. The reputation value can be used in the validation process of job result by using majority voting technique.

Keywords: Grid Computing, Attacks, Tolerance, Threats, Majority Voting, Reputation; Trust.

1 Introduction

A Grid system is a platform that provides access to various computing resources owned by different institutions through the creation of virtual organizations [1]. A grid virtual organization (VO) allows a seamless aggregation of computational re-

Ahmed Bendahmane · Mohammad Essaaidi · Ahmed El Moussaoui · Ali Younes
Information and Telecommunication Systems Laboratory,
Faculty of Science, Tetuan, Morocco
e-mail: `a.dahman@uae.ma, essaaidi@ieee.org, elmoussaoui@uae.ma, younes.ali@uae.ma`

sources in multiple administrative domains into a single pool. The users can lease for a certain time-frame bundles of software, CPU cycles, bandwidth and storage space in order to execute computationally intensive tasks, according to a service level agreement (SLA) between the user and the grid broker service. This broker is acting as a service intermediate between users and grid resources. This brokering role can either happened in strong structured grids, with long-lasting VOs inside the VO boundaries or in volatile grid environments, with very dynamic VOs, where the grid resources can dynamically join and leave in the middle of an execution without impediment. In the second scenario, which is very close to the grid design adopted by the XtreemOS project [15], the execution of a task may be halted arbitrarily. It is then mandatory for VO manager to tolerate these misbehaviors in order to assure the security of the user jobs execution.

Moreover, as the grid resources are connected through the Internet, and with the fusion of web services and grid technologies, and due to the inherently dynamic and complex nature of grid system, a range of new security threats and vulnerabilities are coming to the fore with the broadening scope of grid computing [3], and especially the grid resources has become a subject of several unknown attacks. Introduction of grids in the commercial sector has open up new ventures in the business arena [8]. However, the contemporary grid application domains require many protections from malicious entities. These applications and their underlying infrastructure have to protect the critical information about their owners; provide safeguards to the economic interests of the enterprises using the grid system; and win the confidence of the society which is already doubtful of the digital data processing [5].

Conventional grid security mechanisms can ensure data confidentiality and integrity. The existing attacks/intrusions detection and prevention mechanisms are not efficient against unknown attacks. Therefore, another kind of defense mechanism is needed to implement the precaution against all possible attacks/intrusions. This creates the requirement that grid environments should detects and tolerates the compromised resources attacked by an adversary, which tamper with the computation and return corrupted results, in order to guarantee reliable execution in such a suspicious environment.

In this paper, we propose a new approach for tolerating compromised resources by using majority voting mechanisms, and then investigate the trustworthy of the grid resources which are used in voting procedure. This investigation is based on reputation management system. And the validation of result is decided based on the reputation level of the grid resources.

The remainder of this paper is organized as follow. Section 2 presents our motivation of using sabotage tolerance techniques in grid and survey the background of these techniques. In section 3, we describe our approach of using majority voting with reputation, we present our vision of basic components of grid system and the possibility of attacking these components. Section 4 presents how the grid broker service will build the reputation of grid resources. Section 5 concludes the paper.

2 Motivation and Background

2.1 Motivation

Sabotage tolerance is gaining importance in grid environments notably in the situation where different grid domains have conflicting interests. The term sabotage tolerance was originally coined in the specific area of Desktop Grids, where voluntarily Internet users contribute to the grid with computing cycles. Because everyone can take part in such a grid, the environment started loosing its trustworthiness. Since computations run in an open and un-trustable environment, it is necessary to protect the integrity of data and validate the computation results. Sabotage tolerance techniques need to be employed mainly for the detection of malicious entities who may submit erroneous results.

In a classical grid, sabotage tolerance is not an issue because the grid environment is trustable, in the sense that someone (grid node administrator) controls strictly the grid resources and their ownership. When a grid resource is malfunctioning, the grid owner is notified. However, if the grid scales at the size of the Internet or spans over multiple administrative domains without a hierarchical subordination of control and ownership, an attacker might corrupt the grid resources by exploiting some of its vulnerabilities, then it is likely that grid resources report erroneous job results. Sabotage tolerance becomes mandatory, for the protection of both the grid users and other grid contributors.

Sabotage tolerance techniques are applied in grid systems that employ the master-worker computational model [13]. This grid model is not restrictive and maps well on the wide variety of grids. This model can be summarized as a server (referred further as the master) that distributes work units of an application to grid nodes (workers). A result error is any result returned by a worker that is not the correct value or within the correct range of values [10]. Sabotage-tolerance techniques imply detecting result errors, which are very important as they can undermine long computations that have been executing during weeks or even months by several workers [4]. The error rate ε is defined as the ratio of bad results or errors among the final results accepted at the end of the computation. Thus, for a batch of N work units with error rate ε, the master expects to receive εN errors. For every application, the master employs some sabotage-tolerance mechanism for obtaining an acceptable error rate ε_{acc} with regard to its application. If a grid user comes with an application divided on a big number of tasks (e.g. 10 batches of 100 work units each) and it requires a global error rate of 10^{-2}, the sabotage tolerance technique should provide with a work unit error rate of about 10^{-5} [10]. Many applications, especially the ones from computational biology and physics require bounds on the error rates, as the correctness of the computed results is essential for making accurate scientific conclusions.

2.2 Background

In this section, we provide a short overview for the principal sabotage tolerance techniques in Desktop Grids. Sabotage tolerance techniques for Desktop Grids can be classified in three big classes [4]: replication with voting, sampling and checkpoint-based verification.

Replication with majority voting [13] is widely used in the BOINC Desktop Grid platform [2]. The master distributes $2m - 1$ replicas of a work unit to workers and when it collects m similar results, it accepts that result as being the correct one. Each collected result is seen as a vote in a voting pool with $2m - 1$ voters and with majority agreement being the decision criteria. The error rate of this method is determined by the number of identical results required (m), which is a measure of redundancy. High levels of redundancy provide very low error rates (less than 10^{-5}). The main benefit of the method is its simplicity, while the big drawback is the fact that it wastes a lot of resources.

Sampling-based techniques are developed to overcome limitations of replication, especially redundancy. Within sampling, the master determines the trustworthiness of the workers by verifying them only on a few samples of their results. The basic sampling is the naïve one [14], where the master sends probes with verifiable results to workers. If workers respond well to the probes, they are considered trustworthy. The main drawback is the workers can easily recognize the probes and respond well to them, while cheating on the ordinary work units. The probing by spot checking is introduced by Sarmenta [13]. Probes - named now spotters, are tasks with known results. If a worker fails to compute correctly a spotter, it will get blacklisted and all its results will be invalidated. Based on spot-checking, Sarmenta defines the credibility of a worker and a result. The credibility of a worker is an estimate of the probability the worker will return a correct result. The credibility of a result is the conditional probability that the result originating from a worker will be accepted as correct. Quizzes [16] are an improvement to basic probing. With this method, the master sends to workers batches with work units and it places the probes inside of those batches. Given the actual required error rate, the master can compute the number of quizzes to place inside a batch. A drawback of probing is the fact the master should possess some heuristic in order to generate the probes and make them to resemble with the real work units. Because this is a difficult task, the master can use actual work units as probes [13]. The master verifies (using replication) only a sample of results and if a worker is caught cheating all its previous results are invalidated.

Checkpoint based verification addresses the problems with sequential computations that can be broken in multiple temporal segments $(S_{t1}, S_{t2}, \ldots, S_{tn})$. At the end of each segment a checkpoint is submitted to a stable storage. The checkpoints are stored locally to allow recovering the long tasks from faults. After finishing a task, the worker sends back the result along with a list of hashes for the checkpoints. In the basic checkpoint verification [11], the master randomly selects a checkpoint time S_{ti} for a task and asks the worker to deliver its local checkpoint $C(S_{ti})$. Then the master computes the task from S_{ti} up to the next checkpoint and compares the results with the hash value submitted by the worker for $C(S_{ti+1})$. If the hash verification succeeds, then the worker passed the verification. In the distributed version of the checkpoint verification, the master selects a third worker for verification purposes and let the verifier to compute the checkpoint verification. The distributed version has the advantage of not overloading the master with a lot of verification tasks. The error rate of this method strongly depends on the number of verified checkpoints - i.e. a high percentage of verified checkpoints yield a low error rate, for the cost of increased computation (redundancy) and bandwidth.

3 Our Approach

3.1 Model and Assumption

Fig. 1 shows the basic components of a grid system of interest, which consists of *N* Virtual Organizations (VOs), where each VO contain a set of resources, services and middleware, applications, and users, connected by a local network and the Internet (Fig. 2). From the service point of view, these components are considered as resources provider and services provider. The former owns computational nodes or other physical resources, and has root privilege. It decides whether and when resources should contribute and the quantity of available resources that are needed for job execution. The number of resources is dynamic and undetermined. Besides, each resource provider may require different authentication, authorization mechanisms and security policies. The second is the owner of the software solutions and the information databases that are deployed on a resource providers assets. The software solution is an individual service or a combination of services to solve users submitted jobs or requests. In combination situation, the service provider then comes to be a user that requests other services.

In grid computing environment, a client is a parallel job submitter who requests results, he/she can submit a job to grid broker service (after SLA establishment), which determines where and when to run the job. The broker primarily mediates user access to grid resources by discovering suitable services and resources provided

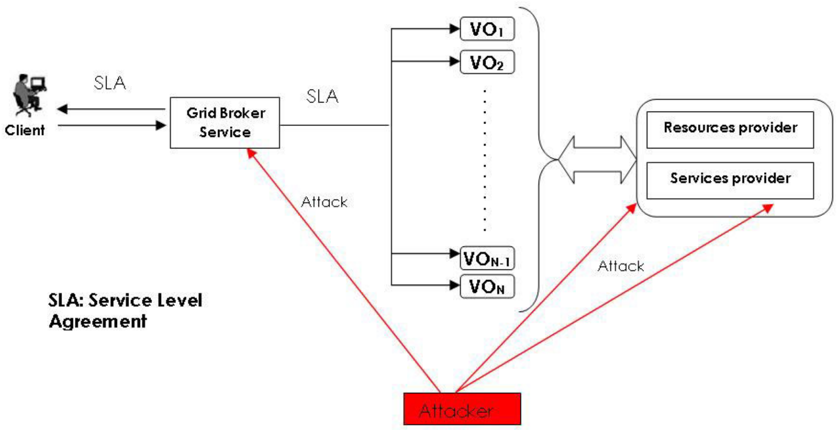


Fig. 1 Basic components of a grid system

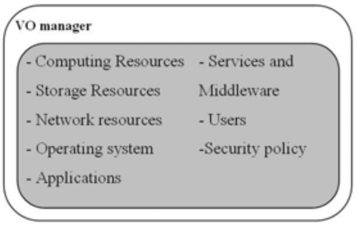


Fig. 2 VO components

by VOs and deploying and monitoring job execution. In the other hand, each VO in the grid has a single VO manager, which searches for available services. When it finds an interesting service, it negotiates with the service provider to allow access to the service from their VO. Once the service provider authorizes the use of the service, the service becomes available at this VO. The job is then divided into sub-jobs that have their own specific input data. The sub-job is called a task. The broker distributes tasks to selected grid resources (computing resources) using scheduling mechanisms. When each grid resource subsequently finishes its task, it returns the result of the task to the broker. Finally, the broker returns the final result of the job back to the client.

In this execution scenario of jobs, there are several possible attacks to grid resources which might tamper with the computation and return corrupted results(grid resources takeover by an unauthorized user can result to malicious acts like corruption of job data, job termination etc.). Such attacks are usually related to security holes of the services and resources provider [9]. Denial of Service (DoS) attacks [7] may also disrupt job completion or temporarily prevent access to job output by cutting a site off the Resource Broker that has delegated the job and is waiting for the output.

3.2 Tolerance of Compromised Grid Resources Using Reputation

Attacks on grid computing, as on any other system, rely on a large number of different vulnerabilities in grid security. Therefore, these attacks become an increasingly prevalent form of security threats, and the problem appears to be quite difficult to solve, especially for attacks which can corrupt the grid resources and make them provide erroneous results of execution. Nevertheless, there exist several defense mechanisms that have been proposed to detect and prevent these attacks, these mechanisms can stop known attacks, but can not defend against new attacks which can happen in grid system.

To this end, tolerance mechanisms for compromised grid resources become necessary to provide a safe result of job execution. As a solution, we propose an approach of tolerating these types of attacks by using sabotage tolerance techniques.

The proposed method is based on reputation to improve the efficiency of majority voting mechanisms. With majority voting the grid broker service (i.e. master) decide about the trustworthiness of a result immediately after having the result of all replicas of a task (i.e. work unit). This approach tolerates a certain number of incorrect results in a vote. However, it does not resist when, for example an adversary gets control through a virus over a large number of computing resources (i.e. workers), and make them return the same result. In our approach, the broker will postpone the decision until the moment it gathers enough information to infer the trustworthiness of the result. To achieve this information, we will use existing reputation lists of different computing resources in VO.

Let us assume that each task is replicated n times and allocated to several computing resources C_i, so that a broker can collect m different results V_j, where i = 1, 2, . . ., n and j = 1, 2, . . ., m. Each computing resource has its reputation value R_i, this reputation is collected by a grid broker service, which contains the reputation lists for all computing resources C_i. The reputation is a value in the range between 0 and 1. In Sect. 4 we will show how the grid broker service will compute the reputation value.

To make a decision about which result V_j is trustworthy (i.e. accepted result), the grid broker service utilizes a majority voting based on reputation decision criteria.

Let the relativity vector $Re(V_j)$ represents The aggregated vote of all computing resources that return the same result.

$$Re(V_j) = [T(V_j,C_1), T(V_j,C_2), \ldots, T(V_j,C_n)] \tag{1}$$

Where $T(V_j,C_i)$ is the relationship between the result V_j and the computing resource C_i. It is calculated as follow.

$$T(V_j,C_i) = \begin{cases} 1 & \text{, if } C_i \text{ compute } V_j \\ 0 & \text{, otherwise} \end{cases} \tag{2}$$

We define the result reputation $RR(V_j)$ of a given result V_j as the summation of reputations of the computing resources returning the result V_j.
For each result V_j :

$$RR(V_j) = \sum_{i=1}^{n} T(V_j,C_i) R_i \tag{3}$$

To make a decision we fixe a positif threshold value $\lambda < 1$ and we pick the maximum of $RR(V_j)$; (j = 1, . . ., m).

$$\begin{cases} if\ \ max(RR(V_j)) > \lambda \sum_{i=1}^{n} R_i & \text{; the result corresponding to this maximum is accepted} \\ Otherwise & \text{; the grid broker service should decide for further replication} \end{cases}$$

In some way, to prevent the possibility that a set of computing resources with a low reputation could undermine the result, we impose the following condition.

$$R_i = \begin{cases} 0 & \text{, if } R_i < \Theta \\ R_i & \text{, otherwise} \end{cases} \tag{4}$$

Where Θ represents the minimum reputation value (see Sect. 4) a computing resource should have for its results to be taken in consideration.

4 Computing Reputation

In this section, we present how the grid broker service will build the reputation value of each computing resource (or grid resource).

The reputation of an entity is an expectation of its behavior based on other entities observations or information about the entities past behavior at a given time [6].The reputation management services are responsible for evaluating the reputation of grid resources, services, and users inside a VO. In our vision, the grid broker service builds the reputation of each computing resource through its credibility.

The credibility represents the likelihood of a particular object of the system to be operating properly [12]. The broker computes the credibility of the computing resource C_i by passing spot checking (see Sect. 2.2). If f is the proportion of compromised computing resources, the credibility of a computer resource which correctly computed k_i spotters (tasks) will be [13]:

$$CR(C_i,k_i) = 1 - \frac{f}{1-f}\frac{1}{k_i e} \tag{5}$$

Where e is the base of the natural logarithm.

After the computing resources pass enough tasks, they succeed to obtain enough credibility Θ to guarantee that their results are correct given the error rate ε_{acc}.

First, we initialize the reputation list of the computing resources. The initial value is no more than the minimum credibility of the computing resource itself.

$$R_i = CR(C_i,k_i) \tag{6}$$

The reputation value in the list is incremented in such a way that if a computing resource returns a validated result by the reputation-based majority voting described above. we propose to consider as a passed spotter each task susccefully validated by the broker, using the reputation-based majority voting. Thus, we increment k_i and compute R_i.

$$\begin{aligned} k_i &= k_i + 1 \\ R_i &= CR(C_i,k_i) \end{aligned} \tag{7}$$

In the same manner, we do decrease the reputation of those computing resources whose result was not validates by the reputation-based majority voting. We decrement k_i and compute R_i.

$$\begin{aligned} k_i &= k_i - 1 \\ R_i &= CR(C_i, k_i) \end{aligned} \tag{8}$$

When a new computing resource joins the computing environment, it gets as an initial reputation 0. Each result produced by the computing resource will enter the reputation-based majority voting competition. If the result is accepted, the reputation of the computing resource will be increased. If not, the spot-checking method is used to compute its reputation.

5 Conclusion

In this work, the proposed approach for tolerating attacks in grid resources is discussed. This approach is based on majority voting mechanisms and reputation management system. The voting result that is generated by the majority voting can be combined with the reputation of grid resources owned by VOs, and then it may be possible to decide on the authenticity of suspicious grid resources. Thus, our tolerance scheme for detecting suspicious grid resources is more accurate and more reliable. And help to improve the security of grid computing system.

References

1. Berman, F., Fox, G.C., Hey, A.J.G.: Grid Computing: Making the Global Infrastructure a Reality. Wiley, Chichester (2003)
2. David, P.A.B.: Boinc: A system for public-resource computing and storage. In: GRID 2004: The Fifth IEEE/ACM International Workshop on Grid Computing, pp. 4–10. IEEE Computer Society, Washington (2004)
3. Demchenko, Y., Gommans, L., de Laat, C., Oudenaarde, B.: Web services and grid security vulnerabilities and threats analysis and model. In: GRID 2005: The 6th IEEE/ACM International Workshop on Grid Computing, pp. 262–267. IEEE Computer Society, Washington (2005)
4. Domingues, P., Sousa, B., Silva, L.M.: Sabotage-tolerance and trust management in Desktop Grid computing. Future Generation Computer System 23(7), 904–912 (2007)
5. Ermisch, J., Gambetta, D.: People's trust: The design of a survey-based experiment. IZA Discussion Papers 2216, Institute for the Study of Labor (2006)
6. Farag, A., Muthucumaru, M.: Towards Trust-Aware Resource Management in Grid Computing Systems. In: CCGRID 2002: 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID 2002), p. 452. IEEE Computer Society, Washington (2002)
7. Galli, P.: DoS attack brings down Sun Grid demo, `http://www.eweek.com/article2/0,1895,1941574,00.asp` (Cited, Mars 2006)
8. Commercial grid solutions. Grid Computing Planet, `http://www.gridcomputingplanet.com/resources/article.php/933781` (Cited, 18 December 2006)

9. Jiancheng, N., Zhishu, L., Zhonghe, G., Jirong, S.: Threat analysis and Prevention for grid and web security services. In: Proc. of Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 526–531 (2007)
10. Kondo, D., Araujo, F., Malecot, P., Domingues, P., Silva, L.M., Fedak, G., Cappello, F.: Characterizing result errors in Internet Desktop Grids. In: Euro-Par 2007. LNCS, vol. 4641, pp. 361–371. Springer, Heidelberg (2007)
11. Monrose, F., Wyckoff, P., Rubin, A.D.: Distributed execution with remote audit. In: The Network and Distributed System Security Symposium, NDSS. The Internet Society, San Diego (1999)
12. Oliveira, A.C., Sampaio, L., Fernandes, S.F., Brasileiro, F.: Adaptive Sabotage-Tolerant Scheduling for Peer-to-Peer Grids. In: LADC 2009: 2009 Fourth Latin-American Symposium on Dependable Computing (LADC 2009), pp. 25–32. IEEE Computer Society, Joao Pessoa (2009)
13. Sarmenta, L.F.G.: Sabotage-tolerance mechanisms for volunteer computing systems. Future Generation Computer Systems 18(4), 561–572 (2002)
14. Wenliang, D., Jing, J., Mangal, M., Murugesan, M.: Uncheatable grid computing. In: ICDCS 2004: The 24th International Conference on Distributed Computing Systems (ICDCS 2004), pp. 4–11. IEEE Computer Society, Washington (2004)
15. Yang, E.Y., Matthews, B., Lakhani, A., Jégou, Y.: Virtual organization management in XtreemOS: an overview. In: Towards Next Generation Grids, Proc. of the CoreGRID Symposium, Rennes, France, Springer, Heidelberg (2007)
16. Zhao, S., Lo, V., GauthierDickey, C.: Result verification and trust-based scheduling in peer-to-peer grids. In: P2P 2005: The 5th IEEE International Conference on Peer-to-Peer Computing (P2P 2005), pp. 31–38. IEEE Computer Society, Washington (2005)

Pervasive Home Security: An Intelligent Domotics Application

Vincenza Carchiolo, Alessandro Longheu, Michele Malgeri,
Giuseppe Mangioni, and Danilo Torrisi

Abstract. The pervasive computing paradigm promotes new applications in several scenarios. Among these, domotics is receiving a considerable attention. This work presents an intelligent and pervasive surveillance system for home and corporate security based on the ZigBee protocol which detects and classifies intrusions discarding false positives, also providing remote control and cameras live streaming. Results of tests in different environments show the effectiveness of the proposed system.

1 Introduction

Smart domotics systems are receiving an increasing attention thanks to the evolution and gradual standardization of technologies as the ZigBee protocol [1], that can be adopted in disparate environments, reducing costs and increasing interoperability with a growing number of intelligent products from multiple vendors (e.g. sensors, ip cameras and so on).

The outcome of this evolution is the deployment of more and more intelligent applications, often based on the pervasive and ubiquitous computing paradigma [2,3], as for instance home care systems [4], or surveillance systems for home or corporate security [5,6]; the number of such applications is however growing at a limited rate due to the current lack of a consolidated standardization [7]. Moreover, the terms *ubiquitous* and *pervasive* are sometimes mixed [8,9], and overlapped with mobile or embedded computing [10]. Actually, the ubiquitous paradigm comes from the frustrating use of personal computers, and the emerging need of redefining technologies in a human-oriented fashion [11] where computers "vanish into the background" [12],

Vincenza Carchiolo · Alessandro Longheu · Michele Malgeri · Giuseppe Mangioni · Danilo Torrisi
Dip. Ingegneria Informatica e delle Telecomunicazioni,
Facoltà di Ingegneria, Università degli Studi di Catania, Italy

whereas the term pervasive instead generally refers to the possibility of accessing information and services anytime from anywhere [13]. The confusion around such terms or the excessive expectation about their promises however did not reduce their importance and the current and future positive impact they will produce on applications and services.

The work presented in this paper falls into this scenario, in particular we propose a surveillance system for home/corporate security that receives intrusion attempts detected by sensors and cameras connected via ZigBee and classifies such intrusions with a customizable algorithm in order to exclude false positive cases (e.g. leaves moved by the wind). Potential alarms can be managed via an iPhone® application that receives alarms, and allows users to use the iPhone as a system remote controller and as a monitoring console to view real-time camera images.

The paper is organized as follows: section 2 introduces the architecture of the system, while in section 3 we describe in detail the application that manages devices, process alarms detection and implements the push notification service. Section 4 shows the calibration that helps false positive intrusion detection as well as the system at work; finally, section 5 presents our conclusions and further works.

2 The Architecture

The goals of the proposed surveillance system are:

1. to detect unauthorized accesses to the perimeter; indeed, whenever a potential intrusion occurs it should be detected using infrared radars and ip cameras.
2. to classify the events detected. The system must be able to distinguish between real intrusions and false positives e.g. due to leaves moved by the wind or to animals This goal is achieved through a customizable algorithm [14] that can be tailored to both indoor and outdoor environments.
3. to send alarms notification and provide remote system control. In particular, once real intrusions are recognized, the system should provide a notification via an iPhone® application so the user is immediately warned; he should also be able to view real-time images from cameras, so real-time countermeasures as siren activation or police action request can be taken.

The proposed system can be logically represented as in figure 1.

The Supervisor is the heart of the system; it manages the network, manages the data coming from devices (cameras and radars) as well as remote iPhone connections and finally processes video images for intrusions detection. The coordinator acts as the interface between the supervisor and the ZigBee network; it is implemented using a custom ZigBee module, manages the low-level network and it can be configured via standard serial interface.

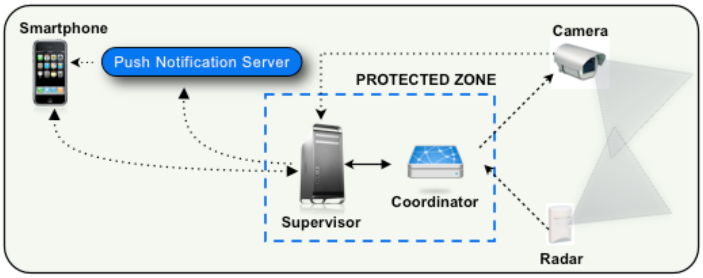


Fig. 1 Logical schema of the proposed suirveillance system

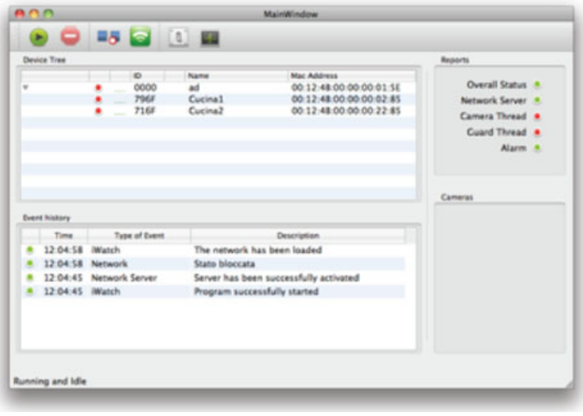
Moreover, we used only devices that provide standard TTL output, in order to easily achieve interoperability with any programmable digital circuit. To manage the communication between the devices and the ZigBee module we used a microcontroller that listens to events from the device and controls the ZigBee module.

The microcontroller waits for signals coming from sensors TTL output; as soon as it receives a signal - i.e. a potential alarm - it alerts the coordinator (via the ZigBee module) that controls the camera using the RS232 protocol. The prototype we have implemented uses a PIC16F628A with an USART (Universal Synchronous/Asynchronous Receiver Transmitter) module that allows a high level management of serial communication. Moreover, the system uses a 16MHz clock in order to ensure the compatibility with the ZigBee module. Note that each module acts as a ZigBee End Device, so they spend most of the time in a sleep state, thus saving energy. The proposed solution offers the advantage that in addition to passive infrared radar it is possible to use any other device (for instance, infrared barrier) as long as its output is TTL logic compliant.

3 Alarms and Their Management

In this section we describe the supervisor, which manages devices and evaluates alarms, helping to prune false positives. The application that implements the supervisor (fig.2) is arranged into separate modules (layers):

- Control module: it controls the physical connection with devices and is based on the QextSerialPort project [15]; this module automatically detects new devices (for instance, a new radar) and allows the supervisor user to add it to the ZigBee network
- ZigBee management module: it is an higher layer manager of the ZigBee devices as cameras and radars;



			ID	Name	Mac Address
▾	🟥		0000	ad	00:12:48:00:00:00:01:5E
	🟥		796F	Cucina1	00:12:48:00:00:00:02:85
	🟥		716F	Cucina2	00:12:48:00:00:00:22:85

Overall Status	🟩
Network Server	🟩
Camera Thread	🟥
Guard Thread	🟥
Alarm	🟩

	Time	Type of Event	Description
🟩	12:04:58	iWatch	The network has been loaded
🟩	12:04:58	Network	Stato bloccata
🟩	12:04:45	Network Server	Server has been successfully activated
🟩	12:04:45	iWatch	Program successfully started

Fig. 2 Snapshot of the supervisor application

- Alarm management module: it processes potential alarms to remove false positives; it was developed in C++ using the Qt Framework [16] and the OpenCV libraries [17]. Moreover, this module manages the remote notification of alarm and control.

In the following (sec. 3.1) we present how the alarms are processed, whereas the subsection 3.2 shows how the remote notification occurs whenever a relevant alarm is detected (low-level details about the ZigBee network are omitted).

3.1 Alarms Processing

The evaluation of alarms is performed by analyzing IP camera images, in particular the MP4 video stream generated by the camera is provided to the supervisor's alarm processing module through a Real Time Streaming Protocol (RTSP) server connected via the wireless connection.

The received frames are evaluated through the algorithm (details can be found in [14]) in order to assign a precision (i.e. a numeric value) thus establishing whether a false positive has been detected. To do this, the first (trivial) solution we adopted was to evaluate the difference between the current frame and the previous one, applying a threshold to detect a binary pattern; this worked in almost static background scenarios, as for indoors environments (e.g. a room) with constant lighting, but it is not suitable when frames background is not static, for instance when tree leaves are moved by the wind. The next step was to apply the Gaussian Background Model [18,19], in order to effectively remove the (even dynamic) background; the algorithm implemented in the supervisor application works following the steps illustrated in fig. 3: it first converts colour images into a black and white format for

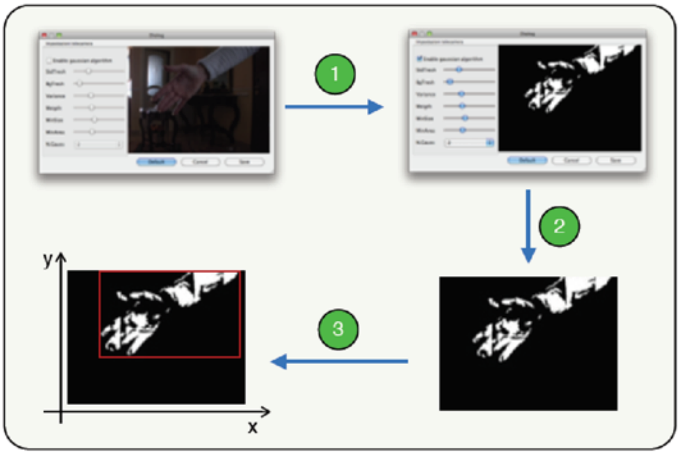


Fig. 3 Steps followed by supervisor algorithm to extract alarms sources

binary processing (1), then it detects the foreground (2) and extracts the corresponding bounding rectangle (3), i.e. the area where the gaussian model detected relevant information.

Bounding rectangles across subsequent frames are compared for a given time interval and if something is detected with a specified precision, an alarm is generated; the system stores in XML format any relevant event, should it determine an alarm or not, in accordance with a set of severity levels. Note that alarm detection is completely customizable by setting algorithm parameters, so a proper configuration allows to correctly distinguish between false positives/negatives and real alarms; for instance, the threshold can be reduced for indoor environments (where the background is almost static) so even a little movement will be detected; similarly, the minimum bounding rectangle can be increased, in order to discard cats/dogs movement detection in outdoor environments and so on. Parameters (whose details are here omitted [14]) should be tailored to the actual scenario the system will be installed into; section 4 shows a typical calibration session.

3.2 Alarm Notifications and Remote Control

As soon as a (possibly real) alarm is detected, the proposed system sends notification to the user, also providing a remote control of overall systems functionalities. These are implemented making use of the well-known push technology [20,21], in particular we used the Apple Push Notification Service (APNS in the following) [22]. The supervisor application is first registered at the server providing APNS, then it will send notification to deliver to the final user by APNS as needed; during our tests, notifications were received within about 4 seconds since the request, that can be considered a good response time for this kind of application.

To manage notifications and provide remote control functionalities, a mobile application operates in conjunction with the supervisor's software counterpart. This application was developed in Objective-C within the Cocoa Touch framework [23], and it displays notifications received from the push server, also allowing the management of the ZigBee network (e.g. enabling or disabling devices) and the possibility of remotely examining current camera video stream from the mobile phone, checking whether it deals with a real intrusion on not and applying proper countermeasures, as an alarm activation or a police call for on-site actions.

4 Experiments and Results

In order to evaluate the performance of the intrusion detection system presented in previous sections, we performed two types of test, the former to assess how the calibration affects alarms claqssification and the latter is about the behaviour of the ZigBee network, in particular for what concerns the largest area that can be effectively kept under surveillance.

To assess false positive alarms pruning, we used a Panasonic^{TM} standard VGA resolution camera in two different scenarios, an indoor environment (a room) with scarce illumination and the an outdoor environment (a garden in front of a house) with strong lighting.

For each scenario we performed 50 test sessions, 25 with an intruder walking in front of the camera and the remaining 25 without intrusion. Finally, results from each scenario have been evaluated first without any configuration for the detection algorithm parameters (i.e. with default values) and then providing calibration for such values (details are here omitted [14]) In the following we present the test results using diagrams that show the number of tests according to the percentage of accuracy reached (ranging from 0 to 100%).

The measures for the indoor scenario with no calibration (see figure 4 a) show unsatisfactory results. In the set of 25 tests with an intruder, 20% of them shown negative results, i.e. the system did not detect the intruder. The same scenario with no intruder shows incorrect results for 44% of tests (a non-existent intruder is detected); this was due to missing calibration and also by changes in the light coming from the windows (which lead to false positive detection).

After the calibration, results significantly improved as figure 4 b shows. The algorithm indeed detected the intruder with high accuracy (no false negative occurred), and similarly in the case of absence of the intruder (no false positive have been detected).

The second scenario concerns an outdoor environment with strong lighting (a garden in front of a house in the morning); results are displayed in

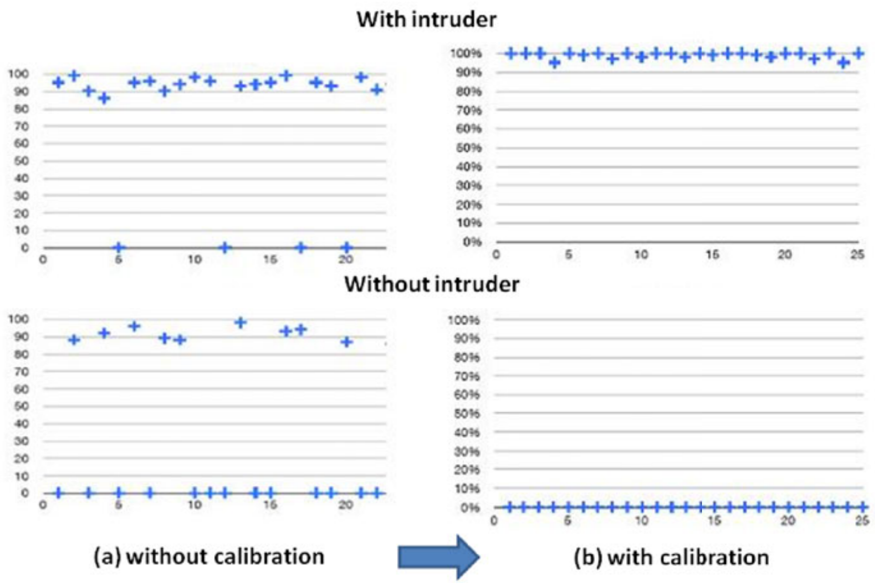


Fig. 4 Indoor scenario

figure 5. The significantly increasing in illumination allow better results even with no calibration (see fig. 5 a); the algorithm indeed shown high precision in detecting an intruder (no false negatives), whilst we have just 16% of false positives in the case of no intrusion. After the calibration results were further improved (false positives were completely prevented).

The second set of tests was about the ZigBee network, in particular we want to assess the largest area that can be effectively kept under suirvelliance. To quantify this effectiveness, we used the Link Quality Indicator (LQI), an 8 bit sequence representing a figure of the quality of the link between two ZigBee nodes [24].

To perfom such tests we used a system made up of a coordinator module and a router module which was positioned at different locations (increasing the distance from the coordination) along a given perimeter. At each position we performed 25 measures; results are shown in fig. 6, where the value of LQI for distance from 0 to 18m is plotted, in particular the solid line represents the theoretical LQI value whereas the other line represents measured LQI values (each point is the average of 25 measures).

Tests revealed worse performances than those declared on the component datasheets, indeed the system worked well up to a distance of 20 meters even if the LQI decreased significantly since a distance of 5 meters. However, we noted that even with a low LQI the communications between ZigBee modules was acceptable, in particular some connection losses were detected but the system was able to restore the connection in few seconds.

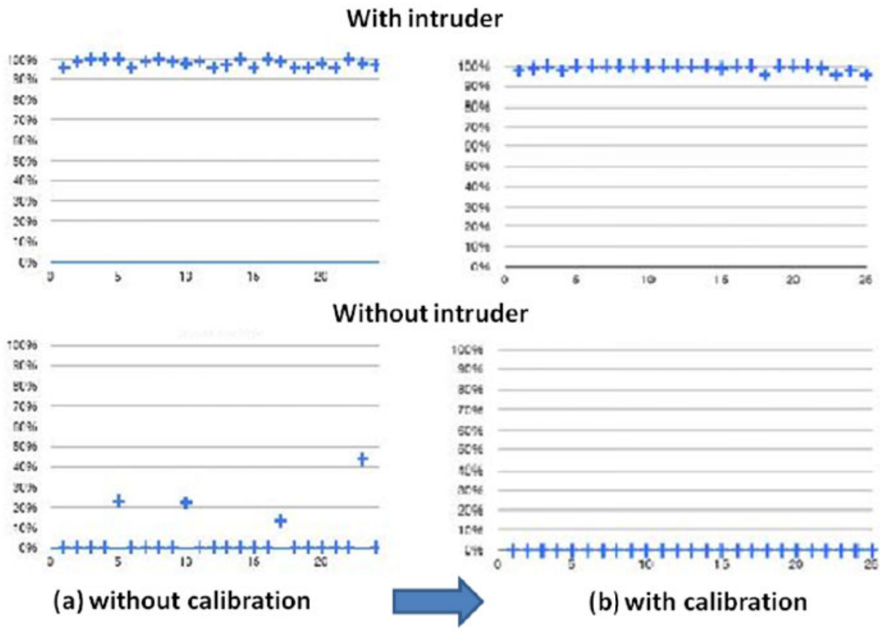


Fig. 5 Outdoor scenario

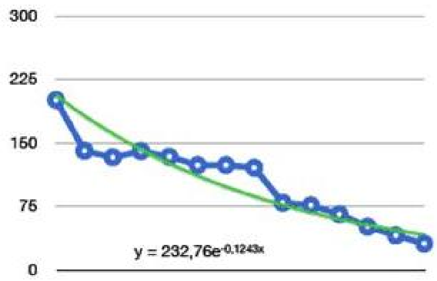


Fig. 6 LQI versus distance between coordinator and router ZigBee modules

5 Conclusions and Future Work

In this paper we introduced an intelligent surveillance system for home (and corporate) security based on the ZigBee protocol capable of detecting and classifying intrusions to discard false positive and negative alarms, also providing remote control functions and cameras live streaming. We also presented results about both the detection algorithm and the ZigBee network performances.

Some future works include the following issues: (1) the application currently was tested on Windows or MacOS X operating systems; to increase portability, we are planning to test it in Linux-based platforms; (2) the system includes just a coordinator and a router with a camera; we planned to add other components as electronic keys or biometric sensors to improve the system's capability. (3) the algorithm we used to evaluate intrusions was quite simple, and it should be intended as the best choice for first experiments (it indeed did not affect on the overall system performance); better yet slower algorithm should be tested.

Acknowledgement. This work has been developed in cooperation with ValueTeam IT consulting and solutions (http://www.valueteam.com). Particularly, we thank Francesco Consoli as Senior Manager at Security Division, for his guidance and support to this work.

References

1. ZigBee Alliance, http://www.zigbee.org/
2. Rodden, T., Benford, S.: The evolution of buildings and implications for the design of ubiquitous domestic environments. In: CHI 2003: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 9–16. ACM, New York (2003)
3. Lorente, S.: Key issues regarding domotic applications. In: International Conference on Information and Communication Technologies: From Theory to Applications, April 2004, pp. 121–122 (2004)
4. Lee, T.-S., Yang, Y., Park, M.-S.: A sms based ubiquitous home care system. In: Yang, L.T., Rana, O.F., Di Martino, B., Dongarra, J. (eds.) HPCC 2005. LNCS, vol. 3726, pp. 1049–1057. Springer, Heidelberg (2005)
5. European Institute for Corporate Security, http://www.eicsm.org/index.html
6. Krausz, B., Herpers, R.: Event detection for video surveillance using an expert system. In: AREA 2008: Proceeding of the 1st ACM workshop on Analysis and retrieval of events/actions and workflows in video streams, pp. 49–56. ACM, New York (2008)
7. Miori, V., Russo, D., Aliberti, M.: Domotic technologies incompatibility becomes user transparent. Commun. ACM 53(1), 153–157 (2010)
8. Nieuwdorp, E.: The pervasive discourse: an analysis. Comput. Entertain. 5(2), 13 (2007)
9. Bell, G., Dourish, P.: Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Personal Ubiquitous Comput. 11(2), 133–143 (2007)
10. McCullough, M.: Digital Ground: Architecture, Pervasive Computing, and Environmental Knowing. MIT Press, Cambridge (2004)
11. Weiser, M., Gold, R., Brown, J.S.: The origins of ubiquitous computing research at parc in the late 1980s. IBM Syst. J. 38(4), 693–696 (1999)
12. Weiser, M.: The computer for the 21st century. SIGMOBILE Mob. Comput. Commun. Rev. 3(3), 3–11 (1999)
13. Hansmann, U., Nicklous, M.S., Stober, T.: Pervasive computing handbook. Springer-Verlag New York, Inc., New York (2001)

14. Torrisi, D.: Sistema di sicurezza perimetrale con allarmistica e controllo a distanza. Technical report, Dipartimento di Ingegneria Informatica e delle Telecomunicazioni - Facolta’ di Ingegneria - Universita’ di Catania (2009)
15. QextSerialPort, http://qextserialport.sourceforge.net/
16. Qt Nokia Framework,
http://qt.nokia.com/about/news/nokia-releases-qt-4.6.2
17. OpenCV libraries, http://sourceforge.net/projects/opencvlibrary/
18. Williams, B.R., Zhang, M.: Multiple dimension chrominance model for background subtraction. In: Computational Intelligence, pp. 438–443 (2005)
19. Lee, D.S.: Effective gaussian mixture learning for video background subtraction. IEEE Trans. Pattern Anal. Mach. Intell. 27(5), 827–832 (2005)
20. Eugster, P.T., Felber, P.A., Guerraoui, R., Kermarrec, A.M.: The many faces of publish/subscribe. ACM Comput. Surv. 35(2), 114–131 (2003)
21. Pohja, M.: Server push with instant messaging. In: SAC 2009: Proceedings of the 2009 ACM symposium on Applied Computing, pp. 653–658. ACM, New York (2009)
22. Apple Push Notification Service,
http://developer.apple.com/IPhone/library/documentation/
NetworkingInternet/Conceptual/RemoteNotificationsPG/
ApplePushService/ApplePushService.html
23. Cocoa Touch Framework, http://developer.apple.com/iphone
24. Orlik, P., Zhang, J., Bhargava, B., Ding, G., Ding, G., Sahinoglu, Z., Sahinoglu, Z.: Reliable broadcast in zigbee networks. In: Proceedings of SECON IEEE Conference (2005)

Ambient Intelligence: From Scenario Analysis towards a Bottom-Up Design

Andrei Olaru, Amal El Fallah Seghrouchni, and Adina Magda Florea

Abstract. In the domain of Ambient Intelligence, research goals are many times driven by scenarios that help envisage a world enriched by ambient, pervasive, intelligent services. So far, scenarios have most times presented the perception that one person has upon the system, with few details on how the system should work in the background in order to deal with realistic requirements. In this paper, starting from scenarios presented in previous research, we identify features and requirements for AmI systems and propose two new scenarios, focusing on the way information is exchanged beyond the perspective of, and transparent to, the user of the system. We also agentify one of the scenarios, giving insight on how an AmI system may be built, using software agents, in order to fulfill the requirements.

1 Introduction

Probably the two best known and most influential papers on related to Ambient Intelligence scenarios and to AmI as a vision came from Weiser in 1995 [9] and from the IST Advisory Group, in 2001 [3]. They describe Ubiquitous Computing and Ambient Intelligence ("AmI", for short): an electronic environment that is present in all things that surround us, as clothes, furniture, objects in our home, vehicles and

Andrei Olaru
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania

Amal El Fallah Seghrouchni
LIP6, University Pierre et Marie Curie, 104 Avenue du President Kennedy,
75016 Paris, France

Adina Magda Florea
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania
e-mail: cs@andreiolaru.ro, amal.elfallah@lip6.fr,
adina@cs.pub.ro

buildings. All things around us (both future "smart" objects and already existing electronic devices) will be connected, web-present [5] and interoperable, will offer intelligent user interfaces and together they will collaborate towards the assistance of people in their daily lives, but also in situations of emergency.

The visions of Weiser and the ISTAG group, together with others [2], describe AmI and its features and challenges with the help of various scenarios. The features of AmI fall into two quite clear categories: intelligent user interfaces (IUI) and an intelligent manner of managing information throughout the system. While most scenarios are centered around the individuals and therefore emphasize the advanced interfaces (like speech recognition, for instance)which is important for the naturalness and invisibility of AmI, we believe that the issue of information management is at least equally important and brings more challenges, not necessarily related to the technologies implied, but mostly related to the internal design and architecture.

In this paper, we briefly analyze scenarios from previous research (Section 2) and identify common features of envisaged or implemented Ambient Intelligence systems, focusing on the layer of AmI concerned with the management of information – above the hardware and network, and below the user interface. We present two new scenarios that emphasize these features, as well as the need for a scaling, dependable and reliable system (Section 3). Finally, we introduce an agentification of one of the scenarios, giving insight on how a distributed, software agent-based AmI system may be built (Section 4).

2 Scenarios for Ambient Intelligence

Weiser's scenario featuring Sal [9] is one of the first scenarios for Ambient Intelligence. A first important thing is that when the alarm clock in the scenario – an intelligent appliance – asks "coffee", the only responses that the clock can interpret are "yes" and "no". That is, the appliance can only understand events that are relevant to its function. Sal's house has intelligent windows that can display traces only of those people that are neighbours, and also considering privacy. Sal marks some news (from a printed paper) with a smart pen, having the associated text sent to her office – the pen is able to contact a service associated with the paper and make the text available in the appropriate context, while keeping Sal's information private. Other services are mentioned, relating to the most common points in AmI scenarios: localization and information on points of interest. Towards the end, Sal succeeds in finding a person (named Mary in the scenario) by means of associations between context data related to the time of the meeting, the place, the number of people, and also to the fact that Sal did not previously know Mary.

The scenarios created by the ISTAG group envision AmI of 2010 [3]. The "Maria" scenario emphasizes the movement of the user's electronic identity, credentials, preferences and data with her, seamlessly, as well as the capacity to easily use local resources (vehicles, utilities, computing and communication capabilities). The "Dimitrios" scenario is based on very advanced digital avatars (D-Me) that can interact naturally with other persons and take context-aware decisions: the senior person's D-Me contacts Dimitrios because he has the same heart condition, and Dimitrios's D-Me finds a child of the same age and situation with his own, for socializing and educational purposes, thanks to common context. The "Carmen" scenario includes a range of services like car pooling, internet shopping, smart fridges, traffic information, vehicle-to-vehicle communication. Finally, in "Anette and Solomon", AmI features natural communication and advanced semantic processing capabilities. What we find interesting in the ISTAG scenarios, apart from the advanced human-machine interfaces, is that most services already exist in the present day, just that in the scenarios the system is capable to provide information and services just in time, they are easier to access and much more context-awareness is present. However, the presented services seem to be centralized – which may not scale well – and they do not seem to have much in common – aggregating information coming from different services may lead to more and better capabilities.

Other scenarios in the Ambient Intelligence field feature context-aware suggestions and files that move between devices to follow the user [7], seamless transfer of network connections, workspace and video conversation to use local devices [1], web-present objects and context-aware activation of suggestions [5], choice of the manner of notification in function of user's context [5, 8].

3 Two New Scenarios

In this section we will present two new scenarios, that give an insight on how an Ambient Intelligence system may work internally. Details are focused on the application layer of the system, on how information is managed and how decisions are taken in function of context. Having this focus, we will consider that users are using today's hardware and connectivity, but devices will be enriched with software agents (that we will call AmI agents), that will form the application layer of the system.

Scenario 1. A senior person walks on the street towards her house. In the pocket she has a mobile phone with an AmI software agent installed, featuring Bluetooth and GSM connectivity and in communication with a multipurpose sensor that monitors vital signs. The AmI agent has been configured to communicate the least possible, so normally it doesn't initiate connections. The person lives in a small basement apartment. As she climbs down the stairs, she falls and loses consciousness for a few moments.

In this short time, the vital signs sensor detects that the situation is not life threatening and no major injury occurred, but care may be needed. The personal medical assistant that cares for this user should be called right away. This reasoning is done by a dedicated module of the AmI agent, that is especially designed for senior or disabled people. As there is no GSM signal at this location, the AmI agent searches for another resource that can offer communication services. It activates Bluetooth and finds a device that also runs an AmI agent. It provides to the other agent some information about the context: there is someone that needs urgent communication by phone. No personal details are provided. The other agent detects that this context fits its activity history: it has helped with this kind of actions before and it accepts the task without the need for owner's confirmation. The agent of the senior person then gives to the other agent a number and a message to be sent. While the senior becomes conscious again, the AmI agent receives, by means of another Bluetooth phone in the area, the confirmation from the nurse. She will arrive in just a few minutes.

Scenario 2. On the largest stadium of an European capital, a concert is going to be held, by a popular rock group of the time. Hundreds of thousands of people are participating. Most of them have mobile phones or smartphones which run AmI agents. Young people are more permeable to new technologies, and the agents are configured to communicate with other agents that share the same context, while keeping personal data private. All participants share the space, time and activity context. AmI agents form a temporary, anonymous social network, communicating not by means of the Internet or by GSM, but by local connectivity like Bluetooth or WiFi ad-hoc networking. They exchange, anonymously, interesting news or links that are related to the event and to the band. The users made that information public and are not necessarily aware of these exchanges, and will view the new data after the concert.

As the concerting band will be an hour late, the organizers send this information to the agents that manage the WiFi access points in the area. In turn, these agents disseminate the information to the devices connected to WiFi. The information is of great relevance to the participants, so it spreads fast among the devices of the people on the stadium. In case other users that are not participating to the event received the information, their AmI agents will discard it because their users are not participating in the event, so the information is not relevant.

Finally, the concert begins. Towards the end, a pyrotechnic event causes a fire on the stage. For security reasons, the public must be evacuated. Panic breaks out. The GSM network soon becomes unavailable and the WiFi hotspots are overloaded. Special emergency devices connect to Bluetooth phones that are located near the exists and send them directions towards the exit. From device to device, the urgent information quickly reaches all participants. AmI agents are capable of calculating the relevance of received information according to the number of links it went through, and choose which exit would be closer.

A few days after the concert, a group of participants that shared a lot of images and links, but not any personal details or contact information, want to find each other again. By using the concert site and the fact that they shared so much, their AmI agents are capable of communicating and the group can meet again.

4 Towards a Model for AmI

AmI must be ubiquitous, pervasive, and it must be capable of managing a large number of mobile devices that incessantly share large quantities of information. More than that, its model must be reliable: in order to be invisible, people must not notice it is there, but also they must not notice when it is not. Considering AmI must be much more than the Internet is today, it must be a *highly distributed* system. Moreover, large quantities of the generated information will only be needed – and, indeed, make sense – inside a certain domain of space, time, activities and social relations (acquaintances). So AmI should work at a *local* level, using *context-awareness* to pick relevant information.

To further illustrate our vision for an Ambient intelligence architecture, we propose a model based on software agents, that uses local interaction and context-awareness.

Software agents [4] are very adequate for the purposes of AmI [6]: they offer pro-activity, reactivity, autonomous behaviour, reasoning, learning and social abilities. In our model, one ore more software agents (called *AmI agents*) run on each device. Each agent is able to connect to AmI agents on other devices in the same area or (across the Internet) with AmI agents that are farther, but that share sufficient context.

Figure 1 presents the interaction between two AmI agents, according to Scenario 1 proposed in Section 3. The two agents are located on the two mobile devices involved. The physical connection is done locally, by Bluetooth, using the connectivity layer. No centralized services are used in the communication. An important role in the conversation is held by context: context information is exchanged between the agents, and each agent evaluates that information and detects context compatibility.

In the other scenario as well, communication and the exchange of information is based on compatible context, and information from incompatible contexts (e.g. related to an event that the user does not participate and is not interested in) is discarded. Local interaction (local in terms of any type of context, not only spatial) makes the system distributed and keeps information stored only there where it is

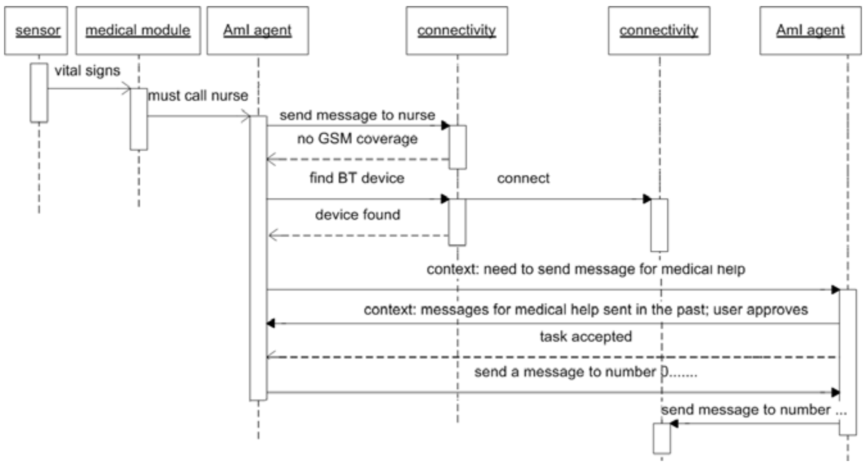

sensor
medical module
AmI agent
connectivity
connectivity
AmI agent
vital signs
must call nurse
send message to nurse
no GSM coverage
find BT device
connect
device found
context: need to send message for medical help
context: messages for medical help sent in the past; user approves
task accepted
send a message to number 0.......
send message to number ...

Fig. 1 Interactions between two AmI agents situated on two different devices (Scenario 1)

relevant. Like in Weiser's scenario, the agents only have the knowledge that may be useful. If the alarm clock is only told "yes" or "no", than it will be only these words that it will understand.

5 Conclusion

Ambient Intelligence is regarded as the third wave in computing and as the future of what is now the Internet. The features of AmI are described many times by means of scenarios which do not emphasize the requirements related to the scale of a real AmI system.

In this paper, we present several scenarios from previous work and we extract some central features and requirements of AmI, that we use for building two new scenarios and sketch a model for AmI that is based on system distribution and local interaction, using the detection of compatible context.

References

1. Banavar, G., Bernstein, A.: Software infrastructure and design challenges for ubiquitous computing applications. Communications of the ACM 45(12), 92–96 (2002)
2. Bohn, J., Coroama, V., Langheinrich, M., Mattern, F., Rohs, M.: Social, economic, and ethical implications of ambient intelligence and ubiquitous computing. In: Ambient intelligence, pp. 5–29 (2005)
3. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Istag scenarios for ambient intelligence in 2010. Tech. rep., Office for Official Publications of the European Communities (2001)
4. Ferber, J.: Multi-agent systems: an introduction to distributed artificial intelligence. Addison-Wesley, Reading (1999)
5. Kindberg, T., Barton, J., Morgan, J., Becker, G., Caswell, D., Debaty, P., Gopal, G., Frid, M., Krishnan, V., Morris, H., Schettino, J., Serra, B.: People, places, things: Web presence for the real world. Mobile Networks and Applications 7(5), 365–376 (2002)
6. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems, 15–18 (2008)
7. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal communications 8(4), 10–17 (2001)
8. Vallée, M., Ramparany, F., Vercouter, L.: A multi-agent system for dynamic service composition in ambient intelligence environments. In: Gellersen, H.-W., Want, R., Schmidt, A. (eds.) PERVASIVE 2005. LNCS, vol. 3468, pp. 1–8. Springer, Heidelberg (2005)
9. Weiser, M.: The computer for the 21st century. Scientific American 272(3), 78–89 (1995)

Equitable Leach-E Protocol for Heterogeneous Wireless Sensor Networks

A. Beni hssane, My. L. Hasnaoui, M. Saadi, S. Benkirane, and M. Laghdir

Abstract. A clustering algorithm is a key technique used to increase the scalability and lifetime of the Wireless Sensor Networks (WSNs). In this paper, we propose and evaluate a distributed energy-efficient clustering algorithm for heterogeneous WSNs, which is called Equitable LEACH-E (ELE). This protocol is an improvement of LEACH-E. In ELE, the cluster-heads are elected by using probabilities based on the ratio between residual energy of each node and the remaining energy of the network. Moreover, it uses a 2-level hierarchy by selecting a cluster head for data transmission. Simulations show that the proposed algorithm increases the lifetime of the whole network and performs better than LEACH and LEACH-E.

Keywords: Wireless Sensor Networks; Clustering Algorithm; Heterogeneous Environment; Energy-Efficient.

1 Introduction

Owing to the advances of Micro-Electro-Mechanical Systems (MEMS) and wireless communication, Wireless Sensor Networks (WSNs) have become an indispensable tool to carry out many applications impossible for other types of networks [1, 2]. A WSN is composed of a large number of sensor nodes that are deployed in ad hoc manner. Clustering technique is used to increase the lifetime of WSNs [3, 4]. In fact, only some nodes are required to transmit data over a long distance and the rest will need to complete short distance transmission only.

A. Beni hssane · My. L. Hasnaoui · M. Saadi · S. Benkirane · M. Laghdir
Chouab Doukkali University, Faculty of Sciences Department of Mathematics and Computer Science, MATIC laboratory, El Jadida, Morocco
e-mail: abenihssane@yahoo.fr, mlhnet2002@yahoo.ca, saadi_mo@yahoo.fr, sabenk1@hotmail.com, laghdirm@yahoo.fr

In [5], it is proposed to elect the cluster-heads according to the energy left in each node. In [6], this clustering protocol is called LEACH-E. Based on LEACH-E protocol, we develop and validate a newer Equitable LEACH-E algorithm called ELE. This protocol is proposed to increase the whole network lifetime on a heterogeneous network with a BS located far away from the sensor area. ELE introduces the 2-level hierarchy concept based on the maximum of the ratio between residual energy and the distance to the BS of each cluster head. This permits a better distribution of the energy load through the sensors in the network.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 exhibits the details and analyzes the properties of ELE. Section 4 evaluates the performance of ELE by simulations and compares it with LEACH and LEACH-E. Finally, Section 5 gives concluding remarks.

2 Related Work

The routing protocols for WSNs can be categorized as follow: the clustering algorithms applied in homogeneous networks are called homogeneous schemes, where all nodes have the same initial energy, such as LEACH [4]; and the clustering algorithms applied in heterogeneous networks are referred to as heterogeneous clustering schemes, where all the nodes of the sensor network are equipped with different amount of energy, such as SEP[7], EECS[8], DEEC[6], and [9]. The EECS protocol elects the cluster-heads with more residual energy through local radio communication. The DEEC protocol is a distributed energy-efficient clustering scheme for heterogeneous wireless sensor networks, in which the cluster-heads are elected by a probability based on the ratio between residual energy of each node and the average energy of the network. In DEEC protocol, the BS is located in the center of the sensing area and uses 1-level hierarchy concept. In [5], a protocol is proposed to elect the cluster-heads according to the energy left in each node. This protocol is called LEACH-E in [6]. The drawbacks of LEACH-E are that it requires the assistance of routing protocol, which should allow each node to know the total energy of network and it utilizes direct transmission from a cluster heads to the BS.

Our work is inspired by the previous approaches, but it differs by introducing to LEACH-E an adapted formulas to estimate the network lifetime, thus avoiding the need of assistance by routing protocol. Since we assume that the BS is far away from the sensing area, we are using 2-level hierarchy concept for transmitting data to the BS.

3 Equitable LEACH-E (ELE)

Our ELE uses the same clustering algorithm, the same strategy in Clusters Head selection, Clusters formation, and Schedule Creation (TDMA)as LEACH-E but differs in Data transmission. ELE algorithm can be summarized as in figure Fig 1.

Since the BS is located far away from the network, the total network energy consumption in each transmission to the BS will be very important. To optimize the energy dissipated in the network, our ELE introduces 2-level hierarchy concept which allows a better use of the energy consumed in the network. In ELE, the probability threshold, that each node s_i uses to determine whether itself to become a cluster-head in each round, is given as follow [6]:

$$T(s_i) = \begin{cases} \frac{p_i}{1-p_i(r mod \frac{1}{p_i})} & , if\ s_i\ \varepsilon\ G \\ 0 & , otherwise \end{cases} \tag{1}$$

where G is the set of nodes that are eligible to be cluster heads at round r. In each round r, when node s_i finds it is eligible to be a cluster head, it will choose a random number between 0 and 1. If the number is less than threshold $T(s_i)$, the node s_i becomes a cluster head during the current round. Also, p_i is defined as follow [5]:

$$p_i(r) = \min\left\{\frac{E_i(r)}{E_{total}(r)}k\ ,\ 1\right\} \tag{2}$$

where $E_i(r)$ is the current energy of node i, k is the desired number of cluster, and $E_{total}(r)$ is an estimate of the remaining energy of the network per round r:

$$E_{total}(r) = E_{initial}\left(1 - \frac{r}{R}\right) \tag{3}$$

where R denotes the total rounds of the network lifetime. The value of R is:

$$R = \frac{E_{initial}}{E_{Round}} \tag{4}$$

where E_{Round}, denotes the total energy dissipated in the network during a round r, is given by:

$$E_{Round} = L[2NE_{elec} + NE_{DA} + (k-1)\varepsilon_{mp}d^4_{toMax\_RatCH} + N\varepsilon_{fs}d^2_{toCH} + E\varepsilon_{mp}d^4_{toBS}] \tag{5}$$

where k is the number of clusters, E_{DA} is the data aggregation cost expended in the cluster-heads, d_{toBS} is the average distance between the cluster-head and the BS, d_{toMax_RatCH} is the average distance between the cluster-heads and the Max_RatCH, which is the CH that has the maximum ratio between residual energy and the distance to the BS of each CHs, d_{toCH} is the average distance between the cluster members and the cluster-head. Assuming that the nodes are uniformly distributed and by using the result in [5, 10], we can get the equations as follow:

$$d_{toCH} = \frac{M}{\sqrt{2k\Pi}} \tag{6}$$

$$d_{toMax\_RatCH} = \frac{1}{M^2}\int\int\sqrt{(x_i+x_j)^2+(y_i+y_j)^2}dxdy \approx \frac{M}{2} \tag{7}$$

$$d_{toBS} = \sqrt{2\Pi}\frac{M}{2} \tag{8}$$

$$k = \sqrt{\frac{\varepsilon_{fs}}{\varepsilon_{mp}}}\sqrt{\frac{N}{2\Pi}}\frac{M}{d^2_{toMax\_RatCH}} \tag{9}$$

Substituting equations (9, 8, 7, 6, 5, 4, 3, and 2) into equation (1), we obtain the probability threshold. Based on the information coordinates and the residual energy included on the message broadcasted, the CHs elected can select one of them which has the Maximum of the Ratio between residual energy and the distance to the BS. We called this intermediate Cluster Head as Max_RatCH. The ratio can be seen as trade off between residual energy and nearness to the BS. The Max_RatCH collects all data coming from all CHs, compress it into a single signal and send it directly to the BS. Each non cluster heads(NCH) sends its data during their allocated transmission time (TDMA) to the respective cluster head. The CH node must keep its receiver on in order to receive all the data from the nodes in the cluster. When all the data is received, the cluster head node performs signal processing functions to compress the data into a single signal. When this phase is completed, each cluster head can send the aggregated data to the Max_RatCH.

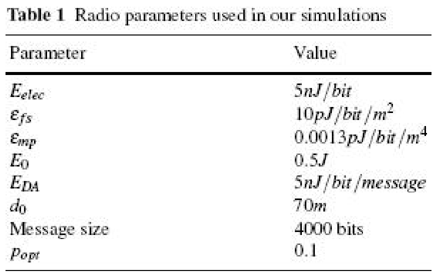
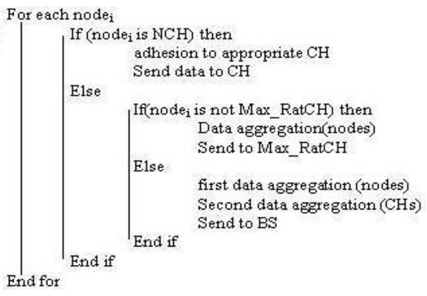


Table 1 Radio parameters used in our simulations

Parameter	Value
E_{elec}	$5nJ/bit$
ε_{fs}	$10pJ/bit/m^2$
ε_{mp}	$0.0013pJ/bit/m^4$
E_0	$0.5J$
E_{DA}	$5nJ/bit/message$
d_0	$70m$
Message size	4000 bits
p_{opt}	0.1



```
For each node_i
    If (node_i is NCH) then
        adhesion to appropriate CH
        Send data to CH
    Else
        If(node_i is not Max_RatCH) then
            Data aggregation(nodes)
            Send to Max_RatCH
        Else
            first data aggregation (nodes)
            Second data aggregation (CHs)
            Send to BS
        End if
    End if
End for
```

Fig. 1 Algorithm

4 Simulation Results

We consider a WSN with N = 100 nodes randomly distributed in a $100m \times 100m$ sensing area. We assume the BS is far away from the sensing region and placed at location $(x = 50, y = 175)$. The radio parameters used in our simulations are shown in Table 1. We assume that all nodes know their location coordinates. We use in this study a similar energy model as proposed in[5]. We will consider the following scenarios and examine several performance measures.

First, we observe the performance of LEACH, LEACH-E, and ELE under two kinds of 2-level heterogeneous networks. Fig. 2 shows the results of the case with $m = 0.1$ and $a = 5$. It is obvious that the stable time of ELE is prolonged compared to that of LEACH and LEACH-E.

Second, we run simulation for our proposed protocol ELE to compute the round of the first node dies when m is varying and compare the results of to LEACH

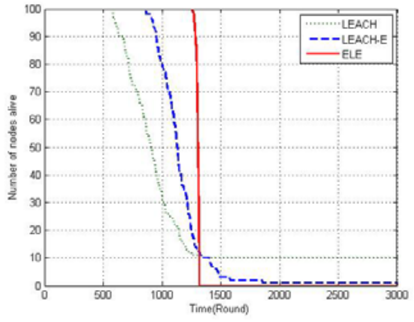


Fig. 2 Number of nodes alive over time ($a = 5$, $m = 0.1$)

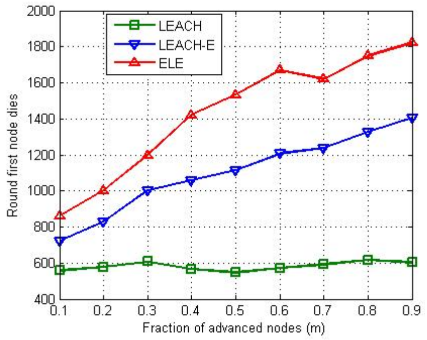


Fig. 3 Round first node dies when m is varying

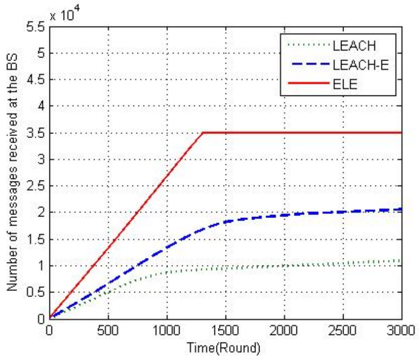


Fig. 4 Number of message received at the BS over time ($a = 5$, $m = 0.1$)

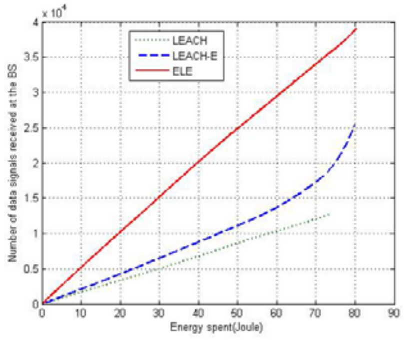


Fig. 5 Number of message received at the BS over energy spent($a = 5$, $m = 0.1$)

and LEACH-E protocols. We increase the fraction m of the advanced nodes from 0.5 to 5, Fig. 3 shows the number of round when the first node dies. LEACH-E performs well and achieves the stability period longer by about 100% than LEACH. ELE outperforms LEACH-E protocol. In fact, when m is varying, ELE obtains 41% number of round than LEACH-E.

Third, Fig. 4 shows that the number of delivered messages to the BS by ELE protocol are greater than the others ones; this means that ELE is a more efficient protocol.

Fourth, Fig. 5 shows that ELE is a more efficient of energy consumption.

5 Conclusion

ELE is proposed as an energy-aware adaptive clustering protocol used in heterogeneous WSNs. To control the energy expenditure of nodes by means of adaptive approach, ELE uses the average energy of the network as the reference energy. Thus, ELE does not require any global knowledge of energy at every election round. Moreover, ELE uses the 2-level hierarchy concept which offers a better use and optimization of the energy dissipated in the network. Finally, the introduced modifications enlarge and outperform better the performances of our ELE protocol than the LEACH and LEACH-E.

References

1. Estrin, D., Girod, L., Pottie, G., Srivastava, M.: Instrumenting the world with wireless sensor networks. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP 2001), Salt Lake City, Utah, USA, May 2001, vol. 4, pp. 2033–2036 (2001)
2. Akyildiz, F., Su, W., Sankarasubramaniam, Y., Cayirici, E.: A survey on sensor networks. IEEE communications magazine 40(8), 102–114 (2002)
3. Mhatre, V., Rosenberg, C., Kofman, D., Mazumdar, R., Shroff, N.: Design of surveillance sensor grids with a lifetime constraint. In: Karl, H., Wolisz, A., Willig, A. (eds.) EWSN 2004. LNCS, vol. 2920, pp. 263–275. Springer, Heidelberg (2004)
4. Heinzelman, W.R., Chandrakasan, A.P., Balakrishnan, H.: Energyefficient communication protocol for wireless microsensor networks. In: Proceedings of the 33rd Hawaii International Conference on System Sciences (HICSS-33) (January 2000)
5. Heinzelman, W.R., Chandrakasan, A.P., Balakrishnan, H.: An application-specific protocol architecture for wireless microsensor networks. IEEE Transactions on Wireless Communications 1(4), 660–670 (2002)
6. Qing, L., Zhu, Q., Wang, M.: Design of a distributed energy-efficient clustering algorithm for heterogeneous wireless sensor networks. Computer Communications 29, 2230–2237 (2006)
7. Smaragdakis, G., Matta, I., Bestavros, A.: SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks. In: Second International Workshop on Sensor and Actor Network Protocols and Applications, SANPA 2004 (2004)
8. Ye, M., Li, C., Chen, G., Wu, J.: EECS: an energy efficient cluster scheme in wireless sensor networks. In: IEEE International Workshop on Strategies for Energy Efficiency in Ad Hoc and Sensor Networks (IEEE IWSEEASN 2005), Phoenix, Arizona, April 7-9 (2005)
9. Liu, M., Cao, J., Chen, G., Wang, X.: An Energy-Aware Routing Protocol in Wireless Sensor Networks. Sensors 9(1), 445–462 (2009)
10. Bandyopadhyay, S., Coyle, E.J.: An energy efficient hierarchical clustering algorithm for wireless sensor networks. In: Proceeding of INFOCOM 2003 (April 2003)

Part VI
Ontologies and Rules

Evaluation of Semi-automated Ontology Instance Migration

Maxim Davidovsky, Vadim Ermolayev,
Wolf-Ekkehard Matzke, and Vyacheslav Tolok

Abstract. Ontology instance migration is one of the challenges in knowledge management. It becomes even more complex in distributed settings when, for example, several autonomous agents use partial assertional knowledge in a domain that is formalized by different though semantically overlapping descriptive theories. Agents exchange instances of their ontologies when cooperate. Such an exchange is essentially the migration of the assertional part of an ontology to other ontologies owned by different agents. The paper presents our method and tool support for migrating instances between different semantically overlapping ontologies. The method is based on the use of manually coded formal rules describing the changes between the input and the output ontologies. The tool to support the process is implemented as a plug-in to Cadence ProjectNavigator software. The main contribution of the paper is in presenting the results of the evaluation of this tool. It reports about the set-up for our evaluation experiments, the metrics used for measuring the quality of instance migration, the ontologies that have been chosen as the experimental data, and the evaluation results. Evaluation results are satisfactory and suggest some directions for the future work.

1 Introduction

Instance migration is an important phase of ontology development and ontology management activities. Until now a large number of ontologies describing

Maxim Davidovsky · Vadim Ermolayev · Vyacheslav Tolok
Zaporozhye National University, Zaporozhye, Ukraine
e-mail: m.davidovsky@gmail.com,
vadim@ermolayev.com, vyacheslav-tolok@yandex.ru

Wolf-Ekkehard Matzke
Cadence Design Systems, GmbH, Feldkirchen, Germany
e-mail: wolf@cadence.com

similar domains from different viewpoints has been developed. Therefore an effective reuse of their assertional knowledge is rational. Ontology instance reuse is also essential in ontology evolution. When a new ontology version is developed it is often necessary to transfer the instances of the previous version(s) to the newer version. The development of a newer ontology version starts with the implementation of the required changes in its TBox. Therefore the reuse of the ABox could be ensured if the elements of the ABox are accordingly transformed. Assertional parts of ontologies can contain a large quantity of instances that in turn makes manual realization of such transformations and instance transfer a laborous task. The paper briefly presents our semi-automated approach to instance migration based on the use of the formal rules describing transformations and transferring instances from a source ontology to a target one. The approach is implemented in our software prototype that has been developed as a plug-in for the Cadence Project Navigator (CPN) Framework for carrying out instance migration between the versions of PSI[1] ontologies [11], [12]. PSI project develops the methodology and the toolset for assessing, predicting, and optimizing the performance of engineering design systems in microelectronics and integrated circuits [10]. A multi-agent system (MAS) for holonic simulation of an engineering design system in the field of microelectronics is developed [20] as a part of the CPN Framework. The MAS assists in analyzing and assessing the performance of a design system as a tool for simulation experiments.

The paper focuses on the evaluation of the approach to instance migration between evolving ontologies in a distributed environment and distributed ontologies with overlapping domains. The scenario of evaluation experiment was developed according to these two use cases. The first phase of the experiment evaluates the quality of instance migration between the versions of PSI ontologies. PSI ontologies are used by cooperating software agents that simulate planning and execution of dynamic engineering design processes [20] in an industrial application domain of Microelectronics and Integrated Circuits. This use case is tailored to support the evolution of the Suite of Ontologies which assertional part is used as the common knowledge of a multi-agent system of cooperative software agents – in distributed settings. The aim of the evaluation based on the second use case is generating reproducible assessment measurements based on the publicly available set of ontologies. For that the benchmark set of ontologies of the Ontology Alignment Evaluation Initiative – (OAEI) 2009 Campaign[2] has been used. The majority of these ontologies are artificially built using the common parent ontology by injecting different sorts of changes. By that a distributed collection of similar ontologies describing the same domain is modeled.

The paper is organized as follows. Section 2 gives a brief overview of the related work. Section 3 describes typical structural change patterns. These

[1] Performance Simulation Initiative (PSI) is an R&D project of Cadence Design Systems GmbH.

[2] `http://oaei.ontologymatching.org/2009/benchmarks/bench.zip`

patterns have been derived from the analysis of the changes in the versions of PSI ontologies and remain valid for the second use case. Change patterns are used in the presented approach for the specification of instance transformation rules. It is explained how instance transformation rules are designed and used for migrating ABox elements. Section 4 presents and discusses the set-up for and the results of the evaluation of the quality of instance migration. Section 5 discusses problems that has been investigated in ontology instance migration and outlines future work.

2 Related Work

When an ontology evolves the structural changes are reflected in the controlled vocabulary. A new TBox version is developed by applying these changes. As a rule TBox development is carried out manually using various ontology engineering methodologies and tools ([3], [4], [13]). The next step of the development of the newer ontology version is transferring the instances from the older ABox to the newer one. Doing this manually proves being very laborious. The effort to be spent appreciably increases with the number of transferable entities. Furthermore it can be error prone.

Very similarly, if distributed ontologies are populated based on the assertional parts of the other ontologies in the same domain[3], the patterns apply for transforming the instances according to the structural differences. However in this case manual transformation is not appropriate as it has to happen at run time. Luckily, different distributed variants of a knowledge representation for the domain may be regarded as different versions of the same ontology that differ by the set of structural changes in the TBox.

The main objective of the approach under evaluation is to develop the tool for semi-automated instance migration from an older version to a newer version of the same ontology. The process of ontology instance migration starts with the analysis of the changes in the TBoxes of the ontology versions. Such an analysis is usually made manually, however there are some software tools that can help in this (e.g. [2]). As a result the rules for transforming the instances based on the patterns for the discovered structural changes are written down by the ontology engineer. These rules are coded in an XML-based[4] language and are further used for guiding the automated transfer of the instances from the older ABox to the newer one. It should be mentioned that the process of making the rules is supported by transformation rules editor that is the part of the developed plugin (Fig. 1). Similar approach is presented in [21]. As opposed to the presented approach, it is based on the use of the ontology instance migration scenario driving the migration process and

[3] For example when cooperating software agents communicate their partial knowledge to their peers in a multi-agent system.

[4] Extensible Markup Language, http://www.w3.org/XML/

encoded as Python scripts. A more detailed problem statement for ontology instance migration and a survey of alternative approaches are given in [5].

A good survey of the related work is [19]. In addition, more frameworks for ontology evolution and ontology versioning need to be mentioned ([16], [15])[5] as they provide some extra bits of the required functionality for ontology instance migration.

Ontology instance migration is the problem that has not been fully researched so far. To the best of our knowledge, the literature reporting the results in this sub-field of ontology management is limited. However, looking at a broader knowledge and data change and transformation landscape is helpful. For instance it is useful taking a look at ontology translation and database transformation fields of research ([14], [1], [7]). Ontology translation approaches can be classified as: ontology extension generation; datasets translation and querying through different ontologies. Dataset translation is of particular relevance to our work.

To the best of our knowledge currently there is no a software tool allowing for effectively carrying out instance migration in a fully automated mode. The analysis of the available literature on the tool support reveals that ontology instance migration is often carried out manually, using a tool for defining the differences between the TBoxes of the source and the target ontologies (e.g. PromptDiff [18]).

Finally, the evaluation of the quality of the results of instance migration and, hence, of the efficiency of the used methods is essential. For quality measurements it is possible to adapt the metrics used in data migration [8] and schema matching [6]. These metrics originate from the information retrieval [17] domain. Our approach to evaluation is based on the use of these adapted metrics.

3 Instance Transformation Patterns and Rules

For correctly migrating instances between a source and target ontology the transfer process has to be explicitly and formally specified. The specification implies an (implicit) declaration of the set of transferable individuals that is achieved by the explicit indication of the set of classes which instances have to be migrated. Also it is necessary to specify the set of required transformations over the migrating individuals. Having done that, we obtain the set of transformation rules for instance migration.

The process of the creation of a new ontology version starts with applying changes to the TBox. Thus comparing TBoxes of respective ontology versions we can identify a certain set of structural changes conditioning differences between the versions. In the case of ontologies having overlapping domains we first have to determine the correspondences between the entities forming the

[5] Also the Karlsruhe Ontology and the Semantic Web tool suite, http://kaon.semanticweb.org/

terminological parts of these ontologies. Then using the obtained mappings we can similarly determine structural changes between the ontologies.

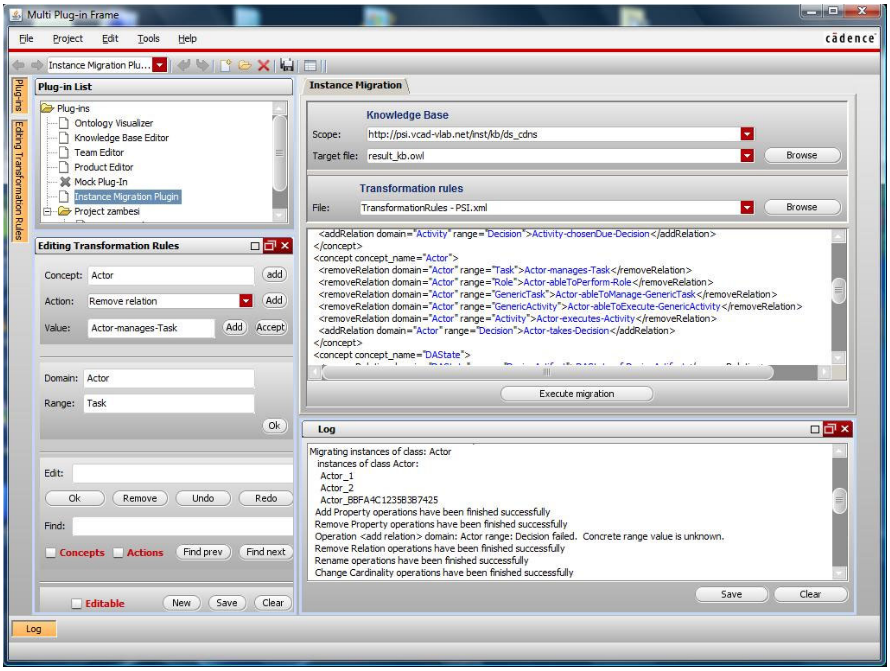
Considering in such a way all possible kinds of changes the set of change patterns that underlie the differences between ABoxes can be defined – in particular such types of changes as the presence or absence of some property, the occurrence of a new relation or the removal of any old relation, etc. Furthermore, based on the set of such change specifications it is possible to define the set of typical transformation operations over the individuals liable to migration. By applying these to concrete ontologies and performing required operations on the instances of the source ontology the target ontology ABox can be obtained.

The analysis of the differences between various versions of PSI ontologies has been done as a part of the presented work. The analysis was carried out as follows. We compared UML diagrams of the ontological contexts [9] of the corresponding concepts and documented the differences. Discovered differences were further cross-checked in the OWL[6] files of the ontologies. For realization of the respective changes between the versions the following transformation patterns have been defined: rename a concept; add a property; remove a property; add a relation; remove a relation; change the cardinality of a relation; change an ancestor (subsumption); and move a concept to another package (ontology module). It should be noted that the operations of moving a concept to another package and subsumption relationship changes do not transform migrated instances. Obviously, referencing of some concept to another package in itself does not entail additional changes and hence will not be reflected in the instances of a concept. Similarly, the fact of referencing some concept as a subclass to another class also does not matter. Probable changes (e.g. inheritance of additional properties) these operations could entail can be described by a combination of the operations and should be discovered when determining differences between ontologies. Therefore the patterns for moving to a different package and changing an ancestor may be not considered. Instance transformation rule patterns can be of two kinds. The first is for the declaration of the set of classes which instances are liable to migration. The second represents typical transformation operations. The patterns are further instantiated in the rule sets for the particular ontologies. These transformation rules are specified by combining appropriate transformation patterns and filling in the pattern slots with concrete values.

Transformation patterns are defined by means of XMLSchema[7]. The use of the patterns provides decoupling from particular ontology contexts and allows producing a set of specific transformation operations for the specific pair of ontologies. In our approach transformation rules are serialized in XML. It allows building convenient and yet computationally efficient formal descriptions for the transformations.

[6] http://www.w3.org/2004/OWL/ - Web Ontology Language.
[7] http://www.w3.org/XML/Schema - XML Schema.

At the step of formulating the transformation rules for a particular migration case the names of the particular classes and the set of transformation operations that should be executed over their individuals are specified in the corresponding XML file. A root element "concepts" in the XML Schema file is defined for that. This root element contains the descriptions of the classes and operations for the particular pair of ontologies. It is formalized as a set of "concept" elements with respective properties that contain as their values the names of classes which instances are liable to migration. The tag <concept> contains the embedded tags that correspond to operations which may also have attributes (for example "add property" tag has "data type" attribute for the added property and "value" attribute that contains the value of the property). The following types of transformation operation patterns are defined: "Add relation" (with optional indication of the domain and range of the corresponding object property); "Remove relation" (also with an option of indicating the respective domain and range); "Change cardinality" (with the indication of a property which cardinality should be changed), "Add property" (with the indication of the data type and value of the added property), "Remove property" and "Rename". The software prototype provides the transformation rule editor (Fig. 1).



Concept: Actor
Action: Remove relation
Value: Actor-manages-Task
Domain: Actor
Range: Task
Edit:
Find:

Scope: http://psi.vcad-vlab.net/inst/kb/ds_cdns
Target file: result_kb.owl
File: TransformationRules - PSI.xml

```
  <addRelation domain="Activity" range="Decision">Activity-chosenDue-Decision</addRelation>
</concept>
<concept concept_name="Actor">
  <removeRelation domain="Actor" range="Task">Actor-manages-Task</removeRelation>
  <removeRelation domain="Actor" range="Role">Actor-ableToPerform-Role</removeRelation>
  <removeRelation domain="Actor" range="GenericTask">Actor-ableToManage-GenericTask</removeRelation>
  <removeRelation domain="Actor" range="GenericActivity">Actor-ableToExecute-GenericActivity</removeRelation>
  <removeRelation domain="Actor" range="Activity">Actor-executes-Activity</removeRelation>
  <addRelation domain="Actor" range="Decision">Actor-takes-Decision</addRelation>
</concept>
<concept concept_name="DAState">
```

```
Migrating instances of class: Actor
  instances of class Actor:
   Actor_1
   Actor_2
   Actor_BBFA4C1235B3B7425
 Add Property operations have been finished successfully
 Remove Property operations have been finished successfully
 Operation <add relation> domain: Actor range: Decision failed.  Concrete range value is unknown.
 Remove Relation operations have been finished successfully
 Rename operations have been finished successfully
 Change Cardinality operations have been finished successfully
```

Fig. 1 Instance Migration Plugin user interface

Let us assume for an example that we do the migration of instances of the concept “Actor” (PSI Actor ontology versions). First, we locate its context in the documentation of changes and identify the following changes in comparison to the previous version of the ontology: (i) the relations with “Task”, “Role”, “GenericTask”, “GenericActivity” and “Activity” concepts are removed; and (ii) the relation with concept “Decision” is added. Hence, the following operations on the instances have to be performed: “Remove a relation” and “Add a relation”. Therefore these operations are created in the editor and written down to the transformation rules file as follows (see also Fig. 1):

```
<concept concept_name="Actor">
  <removeRelation domain="Actor" range="Task">
    Actor-manages-Task</removeRelation>
  <removeRelation domain="Actor" range="Role">
    Actor-ableToPerform-Role</removeRelation>
  <removeRelation domain="Actor" range="GenericTask">
    Actor-ableToManage-GenericTask</removeRelation>
  <removeRelation domain="Actor" range="GenericActivity">
    Actor-ableToExecute-GenericActivity</removeRelation>
  <removeRelation domain="Actor" range="Activity">
    Actor-executes-Activity</removeRelation>
  <addRelation domain="Actor" range="Decision">
    Actor-takes-Decision</addRelation>
</concept>
```

4 Evaluation

The implemented software prototype has been evaluated with respect to the quality and completeness of the executed instance migration. Schematically the experimental set-up is represented in Fig. 2. The results of the migration have been compared to the available target ABoxes for quality and completeness measurement.

The experiment has been run in two phases. Within the first phase the evaluation has been done using the excerpt of the PSI Suite of Ontologies v.2.0 [11] and v.2.2 [12]. The objective of this phase was revealing possible errors at migration run time. The objective of the second phase was to obtain statistically representative and reproducible results. Therefore a broader set of publicly available test case ontologies has been used. The bigger quantity of used ontologies allowed receiving statistically more precise evaluation measurements using various metrics.

Precision and *Recall* metrics have been adopted from Information Retrieval [17] and further adapted for measuring the quality and completeness of ontology instance migration. In our case *Precision (P)* is the fraction of migrated individuals that are relevant. *Recall(R)* is the fraction of relevant

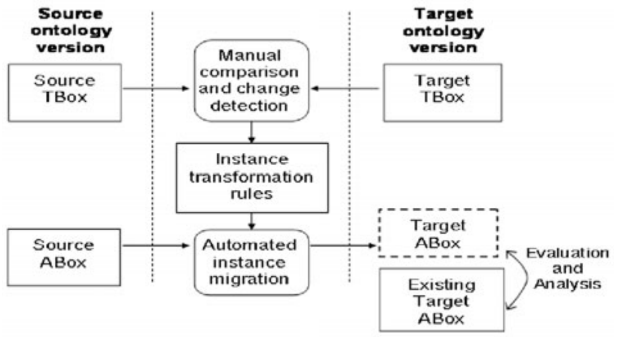


Fig. 2 The set-up of the evaluation experiments

individuals that are migrated. Let's examine the respective contingency Table 1. In the terms of this contingency table $P = tp/(tp + fp)$, $R = tp/(tp + fn)$. The effectiveness of the migration tool can also be measured using the Accuracy metric. In our case *Accuracy (A)* is defined as $A = (tp + tn)/(tp + fp + fn + tn)$. An ideal migration outcome is when *Precision*=*Recall*=1. However neither *Precision* nor *Recall* separately does not provide a complete picture of the correctness of the obtained results. For that the *F measure* could be of interest as it brings *Precision* into correlation with *Recall* as a weighted harmonic mean of both: $F = 1/(\alpha(1/R)+(1-\alpha)(1/R)) = (\beta^2+1)PR/(\beta^2 P+R)$, where $\beta^2 = (1-\alpha)/\alpha$, $\alpha \in [0,1]$. If both precision and recall are of equal importance they can be equally weighted in F by having $\alpha = 1/2$ or $\beta = 1$. This case is the one of a *balanced F measure* $F_{\beta=1} = 2PR/(P+R)$.

Table 1 Instance migration contingency table

	relevant	nonrelevant
migrated	true positives (tp)	false positives (fp)
not migrated	false negatives (fn)	true negatives (tn)

Within the first evaluation phase (the excerpt of the PSI Suite of Ontologies) the total number of instances was 1890. The source for the second phase of the evaluation was the set of the test ontologies of the OAEI 2009. The choice was motivated by the public accessibility of the test set that allows cross-evaluation. 40 ontologies have been chosen which in our opinion were the most appropriable for the instance migration evaluation. The total number of instances was 2060. The contingencies for both evaluation phases are given in Table 2 and the results of the evaluation experiments are summarized in Table 3.

Table 2 The contingency table for different ontology sets

		Relevance	
Test case		relevant	nonrelevant
PSI ontologies	migrated	360	2
	not migrated	48	1480
OAEI ontologies	migrated	1475	7
	not migrated	309	269

Table 3 The summary of the results of the evaluation experiments

	Measures			
Test case	Precision	Recall	Accuracy	Balanced F-measure
PSI ontologies	0.994475138	0.881632653	0.973373303	0.93466032
OAEI ontologies	0.995276653	0.826793722	0.846601942	0.90324556

5 Discussion, Conclusions and Future Work

A number of problems caused by various reasons have been faced in the reported research and implementation work that could be generally described as the lack of the necessary information for transforming all the relevant instances in a correct way. Instance migration problem is therefore substantially under-defined.

The provision of an explicit, correct and complete mapping of a source TBox to a target TBox is inevitably not sufficient for devising the complete and correct transformation of the corresponding assertional parts. A characteristic example is the addition of a new relation to a concept in a target ontology. The values of the respective object property shall be added to the instances of the respective class. However we can not know the exact set of individuals that have to be related. Therefore such a property is not added to all instances in our approach.

A similar collision arises when the cardinality of a relation is changed. We cannot know which particular instances have to be related to the individual having the property with the changed cardinality. However it is known and could be specified in advance that the set of particular instances of a given class has to have the relation to another specific individual. Our approach allows specifying such an a priori knowledge explicitly in the transformation rules. Moreover, at migration run time the log of the migration problems is collected and recorded.

Another sort of migration problems arises because not all required OWL constructs are considered at proper time. A good example case is when the two classes (A and B) in the source ontology are defined as equivalent but they are not equivalent in the target ontology. Based on the equivalence, if the instances of A have been migrated then the instances of B have to be migrated as well. However, non-equivalence in the target ontology may cause collisions at particular individuals. Our approach assumes that all collisions of these sorts have to be explicitly resolved when the changes are discovered and the transformation rules are specified.

Despite that our evaluation experiments proved that the software prototype performs migration correctly and completely if the problems mentioned above are absent or the corresponding collisions are correctly resolved in the transformation rules.

Yet another kind of problems is caused by the imperfection of the transformation procedure, especially at manual steps. If a mistake is made at the step of structural change discovery it will propagate to the transformation rules and further to the automated migration. Moreover, the evaluation metrics for correctness and completeness of the automated migration done by the software are not fully accurate. Fortunately, they can only lower the measures. Our experiments show that even if there were mistakes at manual steps, the results are still quite good. More to mention, the chosen metrics are not invariant to the nature of the test data. For example, *Accuracy* increases if the number of the individuals liable to migration is considerably less than the total number of instances which however does not testify the accuracy of the method. *Precision* and *Recall* do not provide exact reflection on migration quality. High *Precision* can be obtained at the expense of *Recall* by migrating only proper individuals but minimizing the number of migrated instances. Similarly, high *Recall* can be achieved by transferring all relevant instances that inevitably minimize *Precision.*

In a summary it may be stated that the software prototype carries out instance migration correctly on the test cases used in the evaluation experiments and demonstrates sufficiently good results in terms of correctness and completeness. Simplicity, transparency, and validity may be mentioned as the highlights of the developed approach. The method and the prototype software tool allow conveniently creating, editing, and processing instance migration rules. It also reduces the probability of errors at automated processing steps, which is proven by the high value of *Precision* measurements. The lowlights are insufficient flexibility and strong correlation with the correctness of manual specification of transformation rules. The lowlights can be mitigated by offering a refined tool support for manual steps. For instance providing the means for semi-automated structural change detection in the pair of ontologies may lower the chance for manual mistakes at the preparatory steps. Subsequently, automated generation of transformation rules will increase the quality of the proposed approach. Both refinements of the current release of the software prototype are planned for the future work.

Finally it has to be stated that, given the limits of the currently available technologies, it is impossible to completely automate the whole process of ontology instance migration. Therefore, the reduction of the proportion of the manual work is the only feasible way of increasing the performance and quality of results in this important activity of ontology engineering. This observation becomes even more valid in distributed settings. The development of a collaborative platform for ontology engineering teams working on distributed ontologies is one more planned activity for the future work.

References

1. Chalupsky, H.: Ontomorph: A translation system for symbolic logic. In: Proc. Int'l. Con. on Principles of Knowledge Representation and Reasoning, pp. 471–482 (2000)
2. Copylov, A., Ermolayev, V., et al.: Ontology revision visual analysis. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
3. Corcho, O., et al.: Methodologies, tools and languages for building ontologies: where is their meeting point? Data & Knowledge Engineering 46(1), 41–64 (2003)
4. Corcho, O., et al.: Ontological Engineering: Principles, Methods, Tools and Languages. In: Ontologies for Software Engineering and Software Technology. Springer, Heidelberg (2006)
5. Davidovsky, M., Ermolayev, V.: Ontology instance migration. psi. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
6. Do, H., et al.: Comparison of schema matching evaluations. In: Chaudhri, A.B., Jeckle, M., Rahm, E., Unland, R. (eds.) NODe-WS 2002. LNCS, vol. 2593, pp. 221–237. Springer, Heidelberg (2009)
7. Dou, D., et al.: Ontology translation on the semantic web. The Journal on Data Semantics, 35–57 (2005)
8. Drumm, C., et al.: Quickmig - automatic schema matching for data migration projects. In: CIKM (2007)
9. Ermolayev, V., et al.: A strategy for automated meaning negotiation in distributed information retrieval. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 201–215. Springer, Heidelberg (2005)
10. Ermolayev, V., et al.: An upper level ontological model for engineering design performance domain. In: ER 2008: Proc. of the 27th International Conf. on Conceptual Modeling, pp. 98–113. Springer, Heidelberg (2008)
11. Ermolayev, V., et al.: Performance simulation initiative. the suite of ontologies v.2.0. reference specification. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
12. Ermolayev, V., et al.: Performance simulation initiative. the suite of ontologies v.2.2. reference specification. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2009)
13. Fernandez-Lopez, M., et al.: A survey on methodologies for developing, maintaining, evaluating and reengineering ontologies. Tech. Rep. D1.4, OntoWeb Project (2002)
14. Gruber, T.: Ontolingua: A translation approach to providing portable ontology specifcations. Knowledge Acquisition 5, 199–220 (1993)

15. Klein, M., et al.: Ontology versioning and change detection on the web. In: EKAW-2002, pp. 197–212 (2002)
16. Maedche, A., et al.: Managing multiple and distributed ontologies on the semanticweb. VLDB 12, 286–302 (2003)
17. Manning, C.D., et al.: Introduction to Information Retrieval. Cambridge University Press, NY (2008)
18. Noy, N.F., Musen, M.: Promptdiff: A fixed-point algorithm for comparing ontology versions. In: AAAI/IAAI, pp. 744–750 (2002)
19. Serafini, L., Tamilin, A.: Instance migration in heterogeneous ontology environments. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 452–465. Springer, Heidelberg (2007)
20. Sohnius, R., Jentzsch, E., Matzke, W.: Holonic simulation of a design system for performance analysis. In: HoloMAS 2007: Proc. of the 3rd international conference on Industrial Applications of Holonic and Multi-Agent Systems, pp. 447–454. Springer, Heidelberg (2007)
21. Vladimirov, V., Ermolayev, V., et al.: Automated instance migration between evolving ontologies. Tech. rep., VCAD EMEA Cadence Design Systems, GmbH (2007)

Morality in Group Decision Support Systems in Medicine

José Machado, Miguel Miranda, Gabriel Pontes,
António Abelha, and José Neves

Abstract. In this paper we intend to address the morality and behavior problems when using multi-agent systems as a methodology for problem solving when applied to Group Decision Support Systems in Medicine, where the universe of discourse is modeled in terms of a Multi-valued Extended Logic Program (MuvELP) language. It is also presented a formal method to assess moral decisions and moral behavior, accepted as a process of quantification of the Morally Preferable Relation (MPR), measured as the quantity of Quality-of-Information that stems from the logic programs that make the extension of MPR. On the other hand, it was also presented a process to qualify an extension of a logic program as a relational database, therefore taking advantage of the power of relational algebra for problem solving purposes, i.e to set the reason for which something is done or created or for which something exists.

1 Introduction

The healthcare arena is to be understood as a multi-disciplinary environment, where teams with distinct backgrounds and knowledge have to cooperate towards the solution of particular problems and/or to set guidelines for future practices. The notion of teams is not limited to the daily cooperation in each service of an healthcare unit. These teams include elements from different scientific domains and backgrounds. Moreover, considering the moral complications and the complexity of medical practice, argumentation through the use of auxiliary indicators or a multidisciplinary

José Machado · Miguel Miranda · António Abelha · José Neves
University of Minho, Braga, Portugal
e-mail: jmac@di.uminho.pt, miranda@di.uminho.pt, abelha@di.uminho.pt, jneves@di.uminho.pt

Gabriel Pontes
Centro Hospitalar do Alto Ave, Guimarães, Portugal
e-mail: gabrielpontes@chaa.min-saude.pt

team to support the decision making process, presents itself as a mean towards the validation and perhaps improvement of such processes.

The multi-disciplinary teams are intrinsically related with decision support, and information platforms supporting decision support systems are one of the tools with the most potential to adapt and improve such services. Moreover, the organizational and functional schema inherent to an healthcare unit configures a distributed computational environment, where different services and people communicate, exchange data and knowledge (e.g. surgery experiences, patient records, management indicators). This approach to problem solving may be seen as a Multi-Agent System (MAS), where each entity, virtual or human, change arguments with other partakers. Under this happening, each team can intertwine in all healthcare related areas using argumentation and expertise in order to present unbiased plausible explanations, which may be relevant for a particular decision making process. These experts, either human or software agents, presenting their specific knowledge about their expertise area, whenever a requested is made.

On the other hand, Group Decision Support Systems (GDSS) are computer-based systems that aim to support collaborative work, and particularly to increase the benefits and decrease the losses associate to collaborative work; here a meeting is defined as a set of processes necessary to the completion of a specific collaborative task. A meeting is a consequence or an objective of the interaction between two or more entities, which may introduce a moral dimension in all the processes of decision making. Although most of these virtual beings are still rather limited in learning, adaptation and autonomy, displaying solely reactance to predicted or programmed events, several threads of Artificial Intelligence (AI) research methodologies for embedding further intelligence. As virtual entities intervene in decision making processes with moral complications, a justified doubt and concern regarding the impact of actions performed by these entities arises. From the numerous scenarios where they can interact with their surrounding environment, some carry moral consequences and describe ethically intricate actions from a human point of view. From the need to prevent immoral decisions and ensure confidence regarding these virtual entities, further understanding of the capacity of moral agency, moral modeling and the complexity of human moral ethics must be taken into consideration.

2 Multi-agent Based Intelligent Systems in Medicine

The development of an intelligent system is adamantly complex in every area, however in the healthcare arena it has a different context due to the moral value inherent to the human being conditions. All agents (human or virtual) have to be aware of the immeasurable value of an human life and the ethical complexity existing when dealing with this sensible area. From the different paradigms in AI applied to healthcare information systems, Agent Oriented Programming (AOP) has encountered a sheer growth in number and advances. These MAS are capable of implementing distributed autonomous virtual entities, which can integrate other AI techniques such as Genetic Algorithms, Artificial Neural Networks or Case Based Reasoning. An individual agent or a network of agents based on different MAS possess a class of characteristics that allows them to be independent from the will, desires and objectives of other virtual agents, humans or other simple software devices [9]. Although limited, this degree of individuality is the essence of the independence and autonomy to be expected from a moral agent. However, the use of AI learning techniques enables the agents to alter its state and evolve dynamically, making the underlying behavior dependent of the environment itself, which may very well go out of the scope of the initial parameters defined in their process of initialization. This possibility rises issues concerning the ethical and legal responsibility of the initial programmer, in light of the software based agents and of its signature, namely intelligence, autonomy, self-learning and dynamic capabilities, as well as the influence of other external dependencies, which might have influenced the agents in different forms. The use of MAS in Medicine has increasingly occupied an higher level of notoriety, with the introduction of intelligence, pro-activity and efficiency in the healthcare arena. The distributed and heterogeneous nature of this environment makes the best use of this technology [17] [9] [11] [4].

On the other hand, the adoption of an ontology that follows the definitions and requirements for interoperation in healthcare [12] [1], allows the agents to be integrated with the healthcare environment and opens further opportunities to MAS.

3 Principles That Define a Moral Agent and Moral Decision Evaluation

The make up of a moral agency is established by the moral requisites that drive its behavior. Indeed, the fundamental issue that underlies a Moral Agent (MA) is the existence of moral premisses that rule its behavior, differentiating good from evil, being decisive in order to not misunderstand the concept of MA with Moral Patient (MP). While the former has moral obligations, a MP is an entity where moral rights relate to himself or herself. MAs are in most of the cases MPs, however this relation is non-reciprocal, as the discussion on delimitating the grounds of MAs considers that only a part of MPs are in fact capable of being MAs. An adult is a MA although a recently born child is solely a MP, being capable, however, of becoming one during his/her life time [8]. This awareness that an entity may become a MA, is really of great significance, once it allows one to state that an agent, at a given stage of its development process, gets such a property. It is necessary to realize what is a virtual intelligent agent and which characteristics will allow it to become a MA.

According to Wooldridge, an agent embodies a computational system capable of revealing an autonomous and flexible action, developed in a determined universe of speech. The flexibility of the agent is related with the capacity of reaction, initiative, learning and socialization [23]. Although the definition is not universal, for an organic or software based entity, there exits two levels of intrinsic characteristics, which define in a weak or strong form, whether or not in fact that entity is an intelligent agent. On the one hand, the weak notion of agent represents the minim characteristics of an agent, centering on its capacities of autonomy, reactivity, pro-activity and sociability. On the other hand, in the strong notion of an agent, it is defined imminently cognitive characteristics, that may result in the development of a self-consciousness by part of the agent and in the enablement of other valuable properties such as perception, sentimentality, emotions [22]. The establishment of this characteristics is an important factor in the contextualization of the designation of intelligent agents in a way to normalize what is in fact an object, as well as any other form of intelligent software agents.

The comprehension of these characteristics has to be analyzed relatively to a Level of Abstraction (LoA) that uniforms them and limit the possibility of relativism in their analysis. Turing first used the notion of LoA to, according to a level established by him, define intelligence. This concept was used by Floridi and Sander to analyze, according to different LoA the characteristics of intelligent agents, their capacity to undertake moral decisions. Although LoA is a concept used in the Computer Science arena, more concretely from formal methods and uses of discrete mathematics to specify and analyze the behavior of information systems. A LoA consists in a collection of observables, being each one defined by a group of values and results. In other words, to each entity there exists different LoA that characterize it in a distinct way without adding any type of relativity in the resulting observation. Given a set of values X well defined, an observable of the type X is the variable whose response values are contained in X. A LoA consists then in a collection of observable that are considered on the observation. In a less abstract level, in the case of a car, there can be defined different LoA such as of the buyer, mechanic, insurance officer, all of which present different points and characteristics, while being distinct [6].

Depending on the LoA, just as an entity can be considered an object or an intelligent agent, defining the proper LoA the properties that define a MA that may be of use. The three criteria considered in this LoA are interactivity (a), autonomy (b) and adaptability (c), being the synchronous existence of these characteristics what enables an intelligent agent to become a MA [6]. In order to better analyze these characteristics, one must specify and adequate them with the definition of an agent and, as well, with the state of the art in the development of intelligent agents [22].

Modeling moral behaviors in agents is also a field where different agent development methodologies are now being tested, trying to simulate human moral behaviors. Although this simulation may bring a greater understanding of human ethical choices and give a new perspective on moral in general, the lines by which an agent evolves its moral codes are yet to be used as a mean towards building MAs [18].

4 Modeling Moral Behavior in a Multi-valued Extended Logic Programing Setting

There exists no definite solution for modeling ethical virtual entities, and presently several approaches are being presented and some compared against one another. Looking at the present and the ongoing research in this area, different methodologies for modeling moral capabilities using AI techniques can be segmented according to their main characteristics [21]. One of the most definite and important disparity in methodologies is the usage of explicit reasoning versus black-box reasoning. In explicit reasoning, the processes underneath a moral decision are clearly defined as principles, rules, exceptions, or other structure defined for one particular modeling. When analyzing AI techniques derivatives of symbolic, sub-symbolic or statistical approaches, there exist some that are able to represent their line of though, allowing a transparent view of the moral decision process [16]. One of these techniques is based on the potential of logic programming, in which Horn clauses set the formalism that model the reasoning process subsumed by an existing logical predicate. Current research indicates that non-monotic logic, due to its ability to implement defeasible inference, enabling moral principles to add and still diminish the set of conclusions determined by the knowledge base, is an interesting and promising technique to model moral reasoning [16] [19] [7] [3] . Indeed, principles of benevolence and non-malificience can exist in accordance with other principles that are against their value or state an exception for superseding context principles. Regardless of the use of deductive, inductive or abductive logic, the rules used or attained are explicitly defined. However, the usage of each of these techniques of logic programming varies on the objective and context of application.

In our approach, and with respect to the computational model, it were considered extended logic programs with two kinds of negation, classical negation, $\neg$, and default negation, *not*. Intuitively, *not p* is true whenever there is no reason to believe p (close world assumption), whereas $\neg p$ requires a proof of the negated literal. An extended logic program (program, for short) is a finite collection of rules and integrity constraints, standing for all their ground instances, and is given in the form:

$$p \leftarrow p_1 \wedge \ldots \wedge p_n \wedge \text{ not } q_1 \wedge \ldots \wedge \text{ not } q_m; \text{ and}$$

$$?p_1 \wedge \ldots \wedge p_n \wedge \text{ not } q_1 \wedge \ldots \wedge \text{ not } q_m, (n, m \geq 0)$$

where ? is a domain atom denoting falsity, the p_i, q_j, and p are classical ground literals, i.e. either positive atoms or atoms preceded by the classical negation sign $\neg$ [2]. Every program is associated with a set of abducibles. Abducibles may be seen as hypotheses that provide possible solutions or explanations of given queries, being given here in the form of exceptions to the extensions of the predicates that make the program.

Therefore, being Γ a program in Extended Logic Programming (ELP) and $g(X)$ a question where X contains variables $X_1 \wedge \ldots \wedge X_n (n \geq 0)$, one gets as an answer:

The answer of Γ to $g(X)$ is *true* iff
$g(X) \rightarrow demo(\Gamma, g(X), true)$.
The answer of Γ to $g(X)$ is *false* iff
$\neg g(X) \rightarrow demo(\Gamma, g(X), false)$.
The answer of Γ to $g(X)$ is *unknown* iff
not $\neg g(X) \wedge$ not $g(X) \rightarrow demo(\Gamma, g(X), unknown)$.

where *unknown* stands for a truth value in the interval 0...1. Being Γ a Program it is possible to define the Minimal Answer Set of Γ $(MAS(\Gamma))$:

$$\Gamma \vdash s \text{ iff } s \in MAS(\Gamma)$$

where $\Gamma \vdash s$ denotes that s is a logical consequence or conclusion for Γ.

Being now AS_i and AS_j two different answer sets of Γ, being E_{ASi} and E_{ASj}, respectively, the extensions of predicates p in AS_i and AS_j, it is defined that AS_i is morally preferable to AS_j $(AS_i \prec AS_j)$ where $\prec$ denotes the morally preferable symbol, if for each predicate p_1 it exists a predicate p_2 such that $p_1 < p_2$ and $E_{ASi} \setminus E_{ASj}$ is not empty ($\setminus$ denotes the difference set operator).

Case 1

Mr. PD is a man with 81 years, a long background of cardiopathy and diabetes is admitted in an ICU with fever, hypertension and dyspnea. The thorax radiography is compatible with Acute Respiratory Distress Syndrome (ARDS) and the arterial partial oxygen tension (PaO2) is of 50 mmHg. This condition is often fatal, usually requiring mechanical ventilation and although the short-time mortality in these cases has been decreasing, the probability of mortality is considerably high and moreover this procedure results in a low quality-adjusted survival in the first year after ARDS [8, 13]. At the noon service meeting, while analyzing the current cases, the assistant physician asks the interns whether in light of the survival rates, treatment costs and probable low quality of life, should the ICU resources be used with this 81 years old men.

Case 2

During this meeting Mrs. GB, a woman with 36 years interned at the same hospital due to a car accident and diagnosed with sepsis, Acute Lung Injury (ALI) and Glasgow coma scale of 3, shows breathing complications and needs to be admitted an ICU. The level of its PaO2 and the severity of the ALI indicated a pressing need for mechanical ventilation and intensive care. However the number of beds in the ICU is limited and for this matter Mr. PD would have to be changed to another service. Due to the fragile state of Mr. PD this procedure is problematical, but considering his clinical status, complications and age with Mrs. GB, the greater probability of her to full recover with better quality of life tends to tip the balance from a critical point of view.

The Program

The MuvELP for the extension of the agent *survivalrate*:

$\{$
$\neg survivalrate(X,Y) \leftarrow not\ survivalrate(X,Y)$ and
not $exception(survivalrate(X,Y))$,
$exception(survivalrate(X,Y)) \leftarrow survivalrate(X, unknown_{survivalrate})$,
$survivalrate(X,Y) \leftarrow ards(X)$ and $pao2(X,low)$ and $evaluate(X,Y)$,
$exception(survivalrate(gb,0.5))$,
$?((exception(survivalrate(X,Y))$ or
$exception(survivalrate(X,Z)))$ and
$\neg(exception(survivalrate(X,Y))$ and
$exception(survivalrate(X,Z)))$
/This an invariant that states that the exceptions in the extension of the predicate survivalrate follow an exclusive or/
$\}ag_{survivalrate}$

The MuvELP for the extension of agent *survivalquality*:

$\{\ \neg survivalquality(X,Y) \leftarrow$ not $survivalquality(X,Y)$ and
not $exception(survivalquality(X,Y))$,
$exception(survivalquality(X,Y)) \leftarrow survivalrate(X, unknown_{survivalquality})$,
$survivalquality(gb,0.8)$,
$exception(survivalquality(pd,0.1))$, $?((exception(survivalquality(X,Y))$ or
$exception(survivalquality(X,Z)))$ and
$\neg(exception(survivalquality(X,Y))$ and
$exception(survivalquality(X,Z)))$
$\}ag_{survivalquality}$

The Moral Decision

Under this setting, how should the GDSS act?

In our approach, the Morally Preferable Relation (MPR) is based on evolution and it is built on a quantification process of the Quality-of-Information (QoI) that stems from a logic program or theory. Indeed, let $i(i \in \{1,\ldots,m\})$ denotes the predicates whose extensions make an extended logic program or theory that models the universe of discourse, as it is given above for predicates *survivalquality* and *survivalrate*, where $j(j \in \{1,\ldots,n\})$ denote the attributes for those predicates. Let $x_j \in [min_j, max_j]$ be a value for attribute j. To each predicate it is also associated a scoring function $V_{ij}[min_j, max_j] \rightarrow 0\ldots1$, that gives the score predicate i assigns to a value of attribute j in the range of its acceptable values, i.e. its domain (for sake of simplicity, scores are kept in the interval $0\ldots1$). The QoI with respect to a generic predicate it is therefore given by $Q_i = \frac{1}{Card}$, where $Card$ denotes the cardinality of the exception set for the predicate i, if the exception set is not disjoint. If the exception set is disjoint, the QoI is given by $Q_i = \frac{1}{C_1^{card}+\ldots+C_{card}^{card}}$ where C_k^{card} is a

k-combination subset, with *card* elements. The relative importance that a predicate assigns to each of its attributes under observation: $w^i j$ stands for the relevance of the j for the predicate i and is given by $V^i(x) = \sum w^i j = 1$, for all i. The predicate scoring function, i.e. for a value $x = (x_1, \ldots, x_n)$ in the multi-dimensional space defined by the attributes domains, is given in the form $V^i(x) = \sum w^i j V^i j(x)$.

It is now possible to measure the QoI that stems from a logic program or theory, by posting Q_i values into a multi-dimensional space, whose axes denote the logic program predicates with a numbering ranging from 0 (at the center) to 1. The area delimited by the arcs gives a measure of the QoI carried out by the predicate terms used in each problems solution. This was done for the program listed above, therefore defining the process of quantification of the MPR, as expected. In [20], it is shown a new way to represent incomplete information in MuvELP using the relational data model in order to use the power of relational algebra or the Structured Query Language to make inferences, and therefore to present solutions to a particular problem and to measure their degree of self-reliance, as it is the one of Morality in Group Decision Support Systems in Medicine.

5 Conclusion

The first aim of this work was to go through a simple example to present a computational model based on logic to select and measure the QoI of logic programs or theories to solve specific problems of ethics in a group meeting framework, leading to a new methodology for modeling morality in group decision support. Indeed, with the increasing use of Web applications to accomplish the Electronic Health Record (EHR) [15], GDSS obtain the potential to be integrated with other applications in order to support medical and clinical e-learning processes, which will improve the delivery of healthcare and will turn into a fact the provision of life assistance, when and where it is needed. In fact, the usual necessity of documentation and specific information by the medical staff can be easily provided by these systems, making a new step towards a paper free healthcare system. Moreover, this kind of systems allow for a better performance on the part of the tutors, saving time and effort while tutoring the ultimate users. For example, this system has demonstrated great potential in the area of cancer staging [13], the elderly [5] and Intensive Care, where moral dilemmas have a high level of significance. Since we specify a threshold to reduce the combinatory explosion of the solutions, we are discarding non-promising search paths that can be the best route to the results. In future, through implementing convergence methods within creativity programs, in terms of MuvELPs, we will create an evolutionary model to optimize the best theories in terms of their QoI, using the approach presented in this work. Using the evolutionary approach and MuvELP we will be able to predict a program evolution employing the methodologies for problem solving that benefit from abducibles [10] [14].

References

1. Abelha, A., Analide, C., Machado, J., Neves, J., Santos, M., Novais, P.: Ambient intelligence and simulation in health care virtual scenarios. Establishing the Foundation of Collaborative Networks 243, 461–468 (2007), CamarinhaMatos, LM 8th Working Conference on Virtual Enterprises SEP 10-12, 2007 Guimaraes, PORTUGAL
2. Analide, C., Abelha, A., Machado, J., Neves, J.: An agent based approach to the selection dilemma in cbr. In: Badica, C., Mangioni, G., Carchiolo, V., Burdescu, D.D. (eds.) IDC. Studies in Computational Intelligence, vol. 162, pp. 35–44. Springer, Heidelberg (2008)
3. Anderson, M., Anderson, S.L., Armen, C.: An approach to computing ethics. IEEE Intelligent Systems 21(4), 56–63 (2006)
4. Andrade, F., Novais, P., Machado, J., Neves, J.: Contracting agents: legal personality and representation. Artif. Intell. Law 15(4), 357–373 (2007)
5. Costa, A., Novais, P., Costa, R., Machado, J., Neves, J.: A memory assistant for the elderly. Intelligent Distributed Computing III 237, 209–214 (2009); Papadopoulos, G.A., Badica, C.: 3rd International Symposium on Intelligent and Distributed Computing, IDC, Univ. Cyprus, Syia Napa, CYPRUS, October 13-14. Dept. Comp. Sci (2009)
6. Floridi, L., Sanders, J.W.: On the morality of artificial agents. Minds Mach. 4(3), 349–679 (2004)
7. Guarini, M.: Particularism and the classification and reclassification of moral cases. IEEE Intelligent Systems 21(4), 22–28 (2006)
8. Himma, K.E.: Artificial agency, consciousness, and the criteria for moral agency: what properties must an artificial agent have to be a moral agent? In: Ethics and Information Technology. Springer, Heidelberg (2008)
9. Isern, D., Sanchez, D., Moreno, A.: Agents applied in health care: A review. International Journal of Medical Informatics 79(3), 145–166 (2010)
10. Kakas, A.C., Michael, A.: Applications of abductive logic programming. In: IJCSLP, pp. 343–344 (1998)
11. Machado, J., Abelha, A., Novais, P., Neves, J., Neves, J.: Quality of Service in Healthcare Units. In: Bertelle, C and Ayesh, A (ed.) EUROPEAN SIMULATION AND MODELLING CONFERENCE 2008, pp. 291–298. European Simulat Soc; Ghent Univ; Univ Havre; LITIS, EUROSIS, GHENT UNIV, COUPURE LINKS 653, GHENT, B-9000, BELGIUM (2008). European Simulation and Modelling Conference, Havre, FRANCE, OCT 27-29 (2008)
12. Machado, J., Alves, V., Abelha, A., Neves, J.: Ambient intelligence via multiagent systems in the medical arena. Engineering Intelligent Systems for Electrical Engineering and Communications 15(3), 151–157 (2007)
13. Miranda, M.F.M., Abelha, A., Santos, M., Machado, J., Neves, J.: A group decision support system for staging of cancer. In: Weerasinghe, D. (ed.) eHealth, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 1, pp. 114–121. Springer, Heidelberg (2008)
14. Neves, J., Machado, J., Analide, C., Abelha, A., Brito, L.: The halt condition in genetic programming. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) EPIA 2007. LNCS (LNAI), vol. 4874, pp. 160–169. Springer, Heidelberg (2007)
15. Neves, J., Santos, M., Machado, J., Abelha, A.: Electronic health records - organizational, regional, national, or worldwide? In: Long, C., Anninos, P., Pham, T., Anastassopoulos, G., Mastorakis, N. (eds.) 1st WSEAS International Conference on Biomedical Electronics and Biomedical Informatics, pp. 116–121. World Scientific and Engineering Acad. and Soc. (2007)

16. Nugent, C., Cunningham, P.: A case-based explanation system for black-box systems. Artif. Intell. Rev. 24(2), 163–178 (2005)
17. Nwana, H.S.: Software agents: An overview. Knowledge Engineering Review 11(3), 1–40 (1996)
18. Pereira, L.M., Saptawijaya, A.: Modelling morality with prospective logic. In: Neves, J., Santos, M.F., Machado, J.M. (eds.) Progress in Artificial Intelligence, pp. 99–111. Springer, Heidelberg (2007)
19. Powers, T.M.: Prospects for a kantian machine. IEEE Intelligent Systems 21(4), 46–51 (2006)
20. Ribeiro, J., Machado, J., Abelha, A., Neves, J., Fernández-Delgado, M.: Integrating incomplete information into the relational model. In: World Congress on Engineering (2010)
21. Tonkens, R.: A challenge for machine ethics. Minds Mach. 19(3), 421–438 (2009), http://dx.doi.org/10.1007/s11023-009-9159-1
22. Wooldridge, M., Jennings, N.R.: Intelligent agents: Theory and practice. Knowledge Engineering Review 10, 115–152 (1995)
23. Wooldridge, M.J.: Multiagent Systems – A Modern Approach to Distributed Artificial Intelligence, ch. 1. MIT Press, Cambridge (1999)

A Domain Ontology Learning from Web Documents

Ahmed Said Djaanfar, Bouchra Frikh, and Brahim Ouhbi

Abstract. Resources like ontologies are used in a number of applications, including natural language processing, information retrieval(especially from the Internet). Different methodologies have been proposed to build such resources. This paper proposes a methodology to extract information from the Web to built a taxonomy of terms and Web resources for a given domain. Firstly, a (CHIR) method is used to identify candidates terms. Then a similarity measure is introduced to select relevant concepts to build the ontology. The suite of resulting programs, called (CHIRSIM), is easy to implement and can be efficiently integrated into an information retrieval system to help improve the retrieval performance. Experimental results show that the proposed approach can effectively construct a cancer domain ontology from unstructured text documents.

1 Introduction

The goal of a domain ontology is to reduce(or eliminate) the conceptual and terminological confusion among the members of a virtual community of users who need to share electronic documents and information of various kinds. This is achieved by identifying and properly defining a set of relevant concepts that characterize a given application domain. To aid ontology development, numerous methods have been

Ahmed Said Djaanfar
Faculté des Sciences, Dhar El Mahraz-Fès

Bouchra Frikh
B.P. 1796 Atlas Fès, Maroc Ecole Supérieure de Technologie,
Route d'imouzer-Fès, B.P. 2427, Fès, Maroc
e-mail: bfrikh@yahoo.fr

Brahim Ouhbi
Ecole Nationale Supérieure d'Arts et Métiers, Marjane II,
Béni M'Hamed, B.P. 4024, Meknès, Maroc
e-mail: ouhbib@yahoo.co.uk

developed to extract terms from literature automatically. Daille proposed combined techniques to extract terms automatically from corpora by combining linguistic filters and statistical methods [13]. Frantzi et al.[14] developed a C-value/NC-value method to extract multi-word terms automatically. By taking advantage of semantic relations encoded between terms, Grabar and Zweigenbaum [15] developed a two-step approach to collect semantically related terms and to align morphologically linked work forms for term extraction. Ahmad and Rogers [16] to evaluate terms overrepresented in the domain-specific corpus for ontology development. Savova et al.[17] developed a data-driven approach to extract the most specific term for ontology development using an algorithm combining statistical and linguistic approaches. Another tool developed to automatic ontology extraction with text clustering, OntoClust, is given in [19]. These methods focus on extracting terms from the published literature, two other studies also proposed to extract terms from web resources for concept and ontology development [3],[18].

This paper, presents a method and a tool, CHIRSIM, aimed the extraction of domain ontology from the Web. The main contribution of this paper is to present a new methodology for ontology learning from the web to build a taxonomy of terms and web resources for a given domain in two steps. We first introduce a more precise statistic based on an extended version of the χ^2 statistic, called a (CHIR) statistic method [2] to select terms that are positive dependent. This new statistical data can describe the term category dependency more accurately than the statistics used in the paper of Sanchez and Moreno [3], since CHIR keeps only terms relevant to the categories. Moreover, the relations specified in their paper is reduced to the hierarchical relation between concepts. The main assumption in their work is that words that are near to the specified keyword are closely related. So their algorithm is based on the frequency and on the neighborhood of an initial keyword. To relax this hypothesis, we introduce more complex relations between concepts and consequently, the obtained ontology is more rich. According to [2], the CHIR method performs better than three feature selection methods, which are based on the χ^2 statistic and its two variants [2] and references therein. Second, we introduce a mutual information function to measure the similarity between two terms,to remove the terms that are irrelevant or redundant to the task of ontology construction. The mutual information function is then used to acquire linguistic knowledge.This paper is structured as follows. We first present briefly the CHIR-statistic and define a similarity measure to evaluate similarity between two terms (Section 2). In section three, we develop our methodology to extract information from the Web to built a taxonomy of terms and Web resources for a given domain. Section 4, presents the ontology and an application of our algorithm. It is tested on an experimental medical study. Experimental results show that the proposed approach can effectively construct a cancer domain ontology from unstructured text documents. The experimental results show that CHIRSIM allows us to select only terms which are statistically and semantically relevant.

2 Text-Mining Approach to Evaluate Terms for Taxonomy Building

2.1 Term Selection Based on the CHIR Statistic

A number of popular methods have been proposed in textmining, Chi-squared [20], Information gain [21], mutual information [4], etc. A comparison of these techniques can be found in [22]. Recently, Li et al. [2] propose a an extended variant of the χ^2 statistic for term-category independence test to measure the degree of dependency between a term w and a category C of documents. This method can improve the performance of text clustering by selecting the words that help to distinguish the documents into different clusters. Unlike the χ^2 statistic, the main advantage of the CHIR statistic is to select only relevant terms that have strong positive dependency on certain categories in the corpus and remove the irrelevant and redundant terms. Let $O(i,j)$ be the number of documents that are in the category j and contain the term i and $E(i,j)$ be the expected frequency of the category j to contain the term i. The new term-category dependency measure $R_{w,c}$ is defined by:

$$R_{w,c} = \frac{O(w,c)}{E(w,c)}, \tag{1}$$

If there is positive dependency then the observed frequency should be larger than 1. If there is negative dependency, $R_{w,c}$ should be smaller than 1. In the case of the no-dependency between the term w and the category c, the term-category dependency measure $R_{w,c}$ should be close to 1. To get better information about the dependency between a term and a category,Li et al. [2] use a combining formula of $\chi^2_{w,c}$ and $R_{w,c}$ and define the term goodness of a term w in a corpus with m classes as:

$$r_{\chi^2}(w) = \sum_{j=1}^{m} p(R_{w,c_j})\chi^2_{w,c_j} \qquad with \qquad R_{w,c} > 1, \tag{2}$$

where

$$p(R_{w,c_j}) = \frac{R_{w,c_j}}{\sum_{j=1}^{m} R_{w,c_j}} \qquad with \qquad R_{w,c} > 1, \tag{3}$$

is the weight of χ^2_{w,c_j} in the corpus in terms of R_{w,c_j}. A bigger $r_{\chi^2}(w)$ value indicates that the term is more relevant.

2.2 Term Similarity Metric

In the process of term selection, it is desirable to remove the terms that are irrelevant or redundant to the task of ontology construction.The metric we use for measuring the similarity of two terms is that given by Dagan et al. [4]. Two terms are considered similar if their mutual information with all terms in the vocabulary are nearly the same (cf. Brun et al. [7]). The goal of this method is to select a set of terms that maximize the mutual information between the initial keyword and candidate terms. This similarity measure is defined by:

$$Sim(w,w') = \frac{1}{2|V|}\sum_{i=1}^{|V|}\left(\frac{min(I(z_i,w),I(z_i,w')}{max(I(z_i,w),I(z_i,w')}+\frac{min(I(w,z_i),I(w',z_i)}{max(I(w,z_i),I(w',z_i)}\right), \quad (4)$$

where V is the vocabulary and $I(z_i,w)$ is the mutual information of terms z_i and w. In our case, we use it to identify terms which are semantically relevant in a domain ontology construction. This measure is based on the mutual information calculated for a window of d terms. It's nature is essentially semantic than syntactic. The mutual information is defined as follows:

$$I(z_i,w) = P_d(z_i,w)log\frac{P_d(z_i,w)}{d^2P(z_i)P(w)}, \quad (5)$$

where d is the withdrawal, $P(z_i)$ and $p(w)$ are the a priori probability of term z_i and w respectively. $P(z_i,w)$ is the probability of succession of terms z_i and w in the window observation. This probability can be estimated by ratio of the number of times that z_i is followed by w within the window and by the cardinal of the vocabulary.

$$\hat{P}(z_i,w) = \frac{f_d(z_i,w)}{|V|}, \quad (6)$$

where $f_d(z_i,w)$ is the number of times that z_i is followed by w.

3 Taxonomy Building Algorithm

The goal of this section is to construct a taxonomy of terms relevant for a domain. It starts by a keyword which is enough representative of the domain. This hierarchy is built automatically through the algorithm described as follows:

```
Perform a k-means clustering algorithm on the the set of all documents and get
initial clusters.
Start with a keyword that has to be representative enough for the domain and a
set of parameters that constrain the search and the concept selection (cf.[6])
Extract all the candidate concepts by analyzing the neighborhood of the initial
keyword; select the anterior words and posterior words as candidate concepts
For each candidate concept, calculate its $r_{\chi^2}(w)$ statistic by using (2);
Sort the terms in descending order of their $r_{\chi^2}(w)$ statistic.
Select the top $q_1$ terms from the list.
For each term w of a domain, calculate his similarity with the initial keyword
$sim(w,w_{rep})$ as in ;
Sort the terms in descending order of their $sim(w,w_{rep})$.
Select the top $l_1$ terms from the list.
The $l_1$ concepts extracted are incorporated as classes or instances in the taxonomy
and the URLs from where they are extracted are stocked.
```

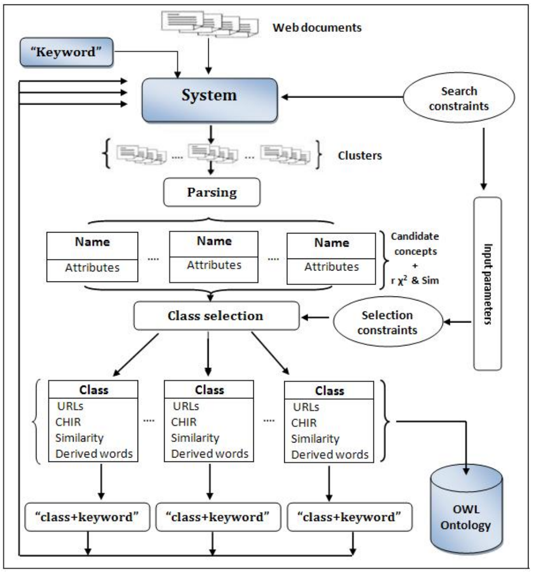


Fig. 1 Taxonomy building methodology

For each concept incorporated in the taxonomy, a new keyword is constructed joining the new concept and the initial one. This process is repeated recursively until a selected depth level is achieved or no more results are found.
Finally, a refinement process is performed in order to obtain a more compact taxonomy and avoid redundancy.

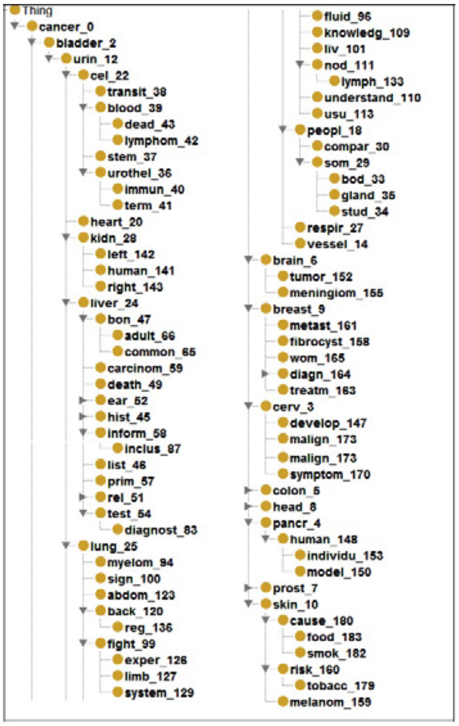
4 Ontology Representation and Application

4.1 Ontology Representation

The final ontology is edited by ”Protégé 4.1” in order to evaluate its precision then it is saved in OWL format. Its content is entirely written in the standard OWL language representation. The Web ontology language OWL was developed as an extension of the RDF (Resource Description Framework) vocabulary and was derived from DAML+OIL Web ontology language (Darpa Agent Markup Language + Ontology Inference Layer). Its main objective is to allow the description of classes and their relations which are inherent to documents and web applications. On the other hand, it is composed in three languages OWL Lite, OWL DL et OWL Full. The final domain ontology (the vocabulary of a domain and a set of constraints on how terms can be combined to model the domain), is presented to the user in a refined way and can be interrogated by the user.

4.2 *Application*

As an illustration application, we choose to use ”cancer” as the initial keyword. The program was executed on a collection of 52 758 documents, aspired from 26 web sites of this domain. The maximum depth level has not been constrained : the system searches subclasses until it finds no more. In the first level, for the CHIR step the depth level is fixed to 15 terms then 9 terms for the SIM step. For other depth levels 18 terms and 11 terms are selected respectively for the CHIR step and the SIM step respectively. The length of the window is fixed to 4. The resulting taxonomy is formally correct. We present in figure 2, a part of the obtained ontology. The numbers in brackets represent the number of instances in the classes.



```
Thing
  cancer_0
    bladder_2
      urin_12
        cel_22
          transit_38
          blood_39
            dead_43
            lymphom_42
          stem_37
          urothel_36
            immun_40
            term_41
        heart_20
        kidn_28
          left_142
          human_141
          right_143
        liver_24
          bon_47
            adult_66
            common_65
          carcinom_59
          death_49
          ear_52
          hist_45
          inform_58
            inclus_87
          list_46
          prim_57
          rel_51
          test_54
            diagnost_83
        lung_25
          myelom_94
          sign_100
          abdom_123
          back_120
            reg_136
          fight_99
            exper_126
            limb_127
            system_129
          fluid_96
          knowledg_109
          liv_101
          nod_111
            lymph_133
          understand_110
          usu_113
        peopl_18
          compar_30
          som_29
            bod_33
            gland_35
            stud_34
        respir_27
        vessel_14
    brain_6
      tumor_152
      meningiom_155
    breast_9
      metast_161
      fibrocyst_158
      wom_165
      diagn_164
      treatm_163
    cerv_3
      develop_147
      malign_173
      malign_173
      symptom_170
    colon_5
    head_8
    pancr_4
      human_148
        individu_153
        model_150
    prost_7
    skin_10
      cause_180
        food_183
        smok_182
      risk_160
        tobacc_179
      melanom_159
```

Fig. 2 Sensor taxonomy visualized on Protégé 4.1

5 Conclusion

This paper, proposes a framework to combine the process of the automatic ontology learning from the Web with the process of information retrieval on the Web. It proposes a methodology to extract information from the Web to built a taxonomy of terms and Web resources for a given domain. Firstly, a (CHIR) method is used to identify candidates terms. Then a similarity measure is introduced to select relevant concepts to build the ontology. The suite of resulting programs, called (CHIRSIM), is easy to implement and can be efficiently integrated into an information retrieval system to help improve the retrieval performance. As an illustration application, an example is given and the resulting taxonomy is formally correct and the obtained results shows that only relevant(statistically and semantically) terms are incorporated in the taxonomy.

References

1. Hartigan, J.A.: Clustering Algorithms. John Wiley & Sons, Chichester (1975)
2. Li, Y., Luo, C., Chung, S.M.: Text Clustering with Feature Selection by Using Statistical Data Knowledge and Data Engineering. IEEE Transactions on Know and Data Eng. 20(5), 641–651 (2008)
3. Sanchez, D., Moreno, A.: Web-scale taxonomy learning, Tech. Rep. of Dep. Computer Science and Mathematics, University Rovira i Virgili (2003)
4. Dagan, I., Lee, L., Pereira, F.C.N.: Similarity-Based Models of Word cooccurrence Probabilities. Machine Learning 34, 43–69 (1999)
5. Church, K., Hanks, P.: Word Association Norms, Mutual Information, and Lexicograph. Computational Linguistics 16(1) (1990)
6. Sanchez, D., Moreno, A.: Creating ontologies from Web documents. In: Recent Advances in Artificial Intelligence Research and Development, vol. 113, pp. 11–18. IOS Press, Amsterdam (2004)
7. Brun, A., Smaili, K., Haton, J.-P.: WSIM: une méthode de détection de thème fondée sur la similarité entre mots. In: 9ème conf. fran. TALN 2002, Nancy, France (2002)
8. Grefenstette, G.: Short Query Linguistic Expansion Techniques: Palliating One-Word Queries by Providing Intermediate Structure to Text. In: Pazienza, M.T. (ed.) SCIE 1997. LNCS, vol. 1299, pp. 97–114. Springer, Heidelberg (1997)
9. Savoy, J., Rasolofo, Y.: Hyperliens et recherche d'information sur le Web, 7eme Journées Internationales d'Analyse statistique des Données Textuelles, JADT 2004 (2004)
10. Sanchez, D., Moreno, A.: Learning non-taxonomic relationships from web documents for domain ontology construction. Data & Knowledge Engineering 64, 600–623 (2008)
11. Alani, H., Kim, S., Millard, D., Eal, M., Hall, W., Lewis, H., Shadbolt, N.: Automatic Ontology-Based Knowledge Extraction from Web Documents. IEEE Intelligent Systems, IEEE Computer Society, 14–21 (2003)
12. Sanchez, D., Moreno, A.: Automatic generation of taxonomies from the WWW. In: Karagiannis, D., Reimer, U. (eds.) PAKM 2004. LNCS (LNAI), vol. 3336, pp. 208–219. Springer, Heidelberg (2004)
13. Daille, B.: Study and implementation of combined techniques for automatic extraction of terminology. In: Resnik, P., Klavans, J. (eds.) The balancing act: combining symbolic and statistical approaches to language, pp. 49–66. MIT Press, Cambridge (1996)
14. Frantzi, K., Ananiadou, S., Mima, H.: Automatic recognition of multi-word terms: the C-value/NC-value method. Int. J. Digit Librar 3(2), 115–130 (2000)
15. Grabar, N., Zweigenbaum, P.: A general method for sifting linguistic knowledge from structured terminologies. In: Proc. AMIA Symp., pp. 310–314 (2000)
16. Ahmad, K., Rogers, M.A.: Corpus linguistics and terminology extraction. In: Wright, S., Budin, G. (eds.) Hand-book of terminology management, vol. 2, pp. 725–760. John Benj. Pub. Co., Amsterdam & Philadelphia (2001)

17. Savova, G.K., Harris, M., Johnson, T., Pakhomov, S.V., Chute, C.G.: A data-driven approach for extracting the most specific term for ontology development. In: AMIA Annu. Symp. Proc., pp. 579–583 (2003)
18. Navigli, R., Velardi, P.: Learning domain ontologies from document warehouses and dedicated web sites. Comput Linguist 30(2), 151–179 (2004)
19. Di Martino, B., Cantiello, P.: Automatic Ontology Extraction with Text Clustering. In: Papadopoulos, G.A., Badica, C. (eds.) Intelligent Distributed Computing III. Studies in Computational Intelligence Series, vol. 237, pp. 215–220. Springer, Heidelberg (2009)
20. Dumais, S.T., Platt, J., Heckerman, D., Sahami, M.: Inductive learning algorithms and representations for text categorization. In: Proceedings of the Seventh International Conference on Information and Knowledge Management (CIKM 1998), Bethesda, MD, USA, pp. 148–155 (1998)
21. Yang, Y., Pedersen, J.P.: A comparative study on feature selection in text categorization. In: Proceedings of the 14th International Conference on Machine Learning (ICML 1997), Nashville, TN, USA, pp. 412–420 (1997)
22. Bakus, J., Kamel, M.S.: Higher order feature selection for text classification. Knowl. Inf. Syst. 9(4), 468–491 (2006)

Articulation and Sharing of Distributed Design Project and Process Knowledge

Vadim Ermolayev, Frank Dengler,
Eyck Jentzsch, and Wolf-Ekkehard Matzke

Abstract. The paper reports on the work in the case study of the ACTIVE project on the use of the combination of knowledge process learning, articulation and sharing technologies for increasing the performance and decreasing the ramp-up efforts in engineering design projects. This knowledge is mined from distributed heterogeneous datasets, fused, and further used for visualizing design project plan information in a way that suggests optimized project plans and fosters collaboration on these knowledge structures in development teams. Software demonstrator architecture, implementation and validation are presented. Validation results indicate that the solution is effective in providing expert assistance to design project managers in performing their typical planning tasks.

1 Introduction

In knowledge economies knowledge workers [4] are central to an organizations success – yet the tools they must use often stand in the way of maximizing their productivity. A need to remedy some of the defects of current knowledge worker tools caused primarily by the requirement for greater knowledge worker productivity becomes substantially more demanded recently and across industries. Knowledge workers acting alone but more importantly in distributed teams are of a particular concern and focus in our research. One of the themes is the support for informal distributed process knowledge acquisition, articulation and sharing.

Vadim Ermolayev · Eyck Jentzsch · Wolf-Ekkehard Matzke
Cadence Design Systems GmbH, Mozartstr. 2 D-85622 Feldkirchen, Germany
e-mail: vadim@ermolayev.com, jentzsch@cadence.com, wolf@cadence.com

Frank Dengler
Institute AIFB, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: frank.dengler@kit.edu

To this theme the notion of an informal (or knowledge) process is central. The definition in [19] lays out the ground for the specific features: Informal processes are carried out by knowledge workers with their skills, experience and knowledge, often to perform difficult tasks which require complex, informal decisions among multiple possible strategies to fulfill specific goals. In contrast to business processes which are formal, standardized, and repeatable, knowledge processes are often not even written down, let alone defined formally, vary from person to person to achieve the same objective, and are often not repeatable. Knowledge workers create informal processes on the fly in many situations of their daily work.

ACTIVE[1] has adopted a service-oriented and component-based approach to its architecture. Services and components are defined at a number of levels [19]. At the bottom level are infrastructure services. At the level above this, machine intelligence technology is used. For example the process mining service learns repeated sequences of action executions which constitute running processes and populates the knowledge base with these (Fig. 1). Finally at the top level are the applications. One of the case study applications is management of design project (DP) knowledge in microelectronic and integrated circuit (MIC) engineering design. This case study is lead by Cadence Design Systems GmbH[2], an engineering design services provider in this domain. Cadence attempts the assessment and management of engineering design performance. This work builds on the results of PSI[3] and PRODUKTIV+[4] projects and goes further, beyond the existing performance management solutions by providing the functionalities of the following two kinds: (i) design process knowledge learning from distributed datasets for knowledge acquisition at the back-end; and (ii) design project knowledge articulation and sharing at the front-end – by providing a tailored lightweight collaboration platform.

Using visualized representations of artifacts in general and of processes in particular is natural for a human and requires less effort for the comprehension [1] of the logic of a process in our case.

Most process visualization techniques are included in process modeling activities, which can be centralized or decentralized. An abundance of modeling methods and tools like ARIS [16] and IDEF3 [13] have been developed to ease the standardization, storage and sharing of process visualization. Unfortunately these tools are not sufficient for modeling collaborative, decentralized processes. Therefore other approaches like CPM [15] have been introduced. In

[1] This work has been funded as part of the IST-2007-215040 EU project ACTIVE. ACTIVE – Enabling the Knowledge Powered Enterprise (http://active-project.eu/) is the EU FP7 Integrating Project.

[2] http://www.cadence-europe.com/

[3] Performance Simulation Initiative (PSI) is the research and development project of Cadence Design Systems GmbH.

[4] Reference System for Measuring Design Productivity of Nanoelectronic Systems (PRODUKTIV+) project has been partially funded by BMBF.

the area of knowledge processes additional methods and tools like KMDL [12], PROMOTE [20] and CommonKADS [17] have been developed extending the methods and tools mentioned above. In addition to that semantic wikis enable large-scale and inter-departmental collaboration on knowledge structures. Such features of semantic wikis have been extended to support process development [3] and workflows [2]. Our approach to knowledge process visualization builds upon our work on dynamic knowledge process representation projects [8]. In this our contribution is the development of the lightweight knowledge process representation for engineering design that is essentially a micro-ontology [5]. The other contribution is in process visualization – the enhancement of the existing Semantic MediaWiki (SMW) process development approach [3] for visualizing and discussing knowledge processes.

2 A Solution for Design Project Knowledge Management

In the domain of MIC engineering design a design system is denoted [10] as a holonic system[5] providing the environment in which design processes are performed. Fig. 1 presents two particular design systems as distributed and heterogeneous environments in the bottom – one for the EU design systems based on the use of the ICD WorkBench framework and the other for the US design systems based on the use of Common Flow Initiative framework. These environments comprise (and differ in) actors, normative regulations and policies, resources, and tools. A design process executed in these environments and the environments themselves are also controlled by performance management processes comprising knowledge acquisition, measurement and assessment, analysis, decision making, and action phases [10]. A software tool for knowledge acquisition, fusion, assessment and analysis facilitates making informed decisions leading to actions that improve the performance of design systems. In the presented case study the focus is on the use and customization of the ACTIVE technologies of pro-active knowledge process support and knowledge articulation and sharing as a part of this decision making support.

There are many aspects related to engineering design process performance that require optimization [9]. One important facet is the dynamic formation of a process. Another one is the requirement to seek for the best productive process continuation among the many possible alternatives. Each execution of such a process is developed by the team of design engineers, design support engineers and project managers in a unique fashion. It is not a challenging problem for experienced team members to make efficient follow-up decisions because they use their tacit knowledge based on experience. However, for those whose experience is limited, an expert adviser may substantially lower

[5] In philosophy a *holon* is a system (phenomenon) that is a whole in itself as well as a part of a larger system (phenomenon).

the risk of a mistake. It is very similar to using a navigator system for finding a route on a map, but on a map displaying design technology stages. In parallel to ACTIVE Cadence is developing such a software system for monitoring, evaluating and simulating design processes for suggesting project continuations with better performance – the ProjectNavigator [18]. At the back-end the ProjectNavigator provides a multi-agent system that simulates the execution of the DP plans by development teams. At the front-end the tool facilitates a DP manager in project planning providing intelligent support based on the knowledge incrementally collected in the Cadence Knowledge Base. This knowledge base and the tool itself are designed using the Suite of PSI Ontologies (e.g. [8, 7])[6].

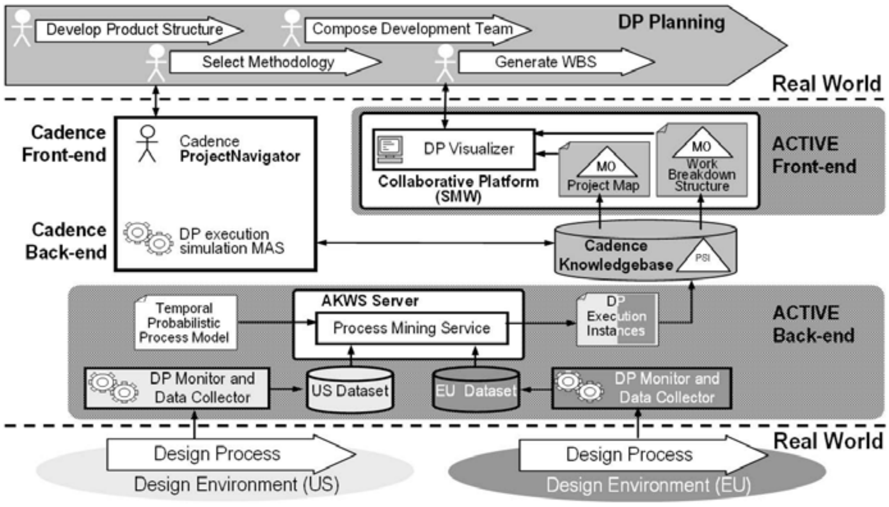


Fig. 1 The configuration of ACTIVE and Cadence technology components for DP knowledge acquisition, articulation and sharing

ACTIVE technologies and software components (Fig. 1) are complementary and facilitating to the ProjectNavigator functionality in several important aspects: (i) at the back-end – for solving the problem of design process execution knowledge acquisition and fusion; and (ii) at the front-end – for solving the problem of DP knowledge articulation and sharing.

Acquisition is done by incremental collection of the new knowledge about the executions of design processes through monitoring design processes and mining the dataset containing design process execution logs – using ACTIVE Process Mining component based on the probabilistic temporal

[6] The documentation for the upper level of the Suite is also available as a Wiki, http://www.kit.znu.edu.ua/mediawiki/index.php?title=PSI_Upper-Level_ontology

process model [11]. The approach of process mining is based on the generation of the Hidden Markov Models [14]. The ACTIVated Design Framework tools monitor the design systems and the design processes and collect the data about the low level events in the respective datasets. The datasets are further fed to the Process Mining Service of the ACTIVE Knowledge WorkSpace (AKWS) Server that produces the instances of the chunks of the executed design processes in terms of the PSI Suite of Ontologies. These instances are further stored to the Cadence Knowledge Base. The fusion of the knowledge coming from distributed and heterogeneous datasets is obtained by applying the same process mining model and the same set of ontology namespaces.

Articulation and sharing are done by visualizing different facets of DP knowledge in the collaborative front-end platform using SMW – the ACTIVE DP Visualizer. Visualization functionality is structured around the typical tasks that DP managers perform in their everyday business (upper part of Fig. 1). The kinds of visualization pages are those for: product structures; generic methodologies; product-bound methodologies; tools; and actor roles. These primary functionalities are supported by decision making instruments for conducting moderated discussions using LiquidThreads[7] extension of MediaWiki or, additionally, LiveNetLife[8]. ACTIVE DP Visualizer is tailored to the specific requirements of the case study. It comprises a software connector that passes the knowledge stored in the Cadence Knowledge Base into the SMW pages (Fig. 1).

The facts are taken from the Cadence Knowledge Base that is the repository of instances of the PSI Suite of Ontologies. The knowledge base is populated from a number of different sources. Static assertions: IP library, methodology, tool, and designer skill descriptions, are entered by human users via ProjectNavigator front-end. Process execution knowledge for the parts of the processes that have already been accomplished comes from ACTIVE process mining components. Predictions about possible continuations of the processes are generated by the ProjectNavigator back-end simulator functionality. The static knowledge, when articulated in the wiki pages, forms the basic map of the design system terrain. The mined knowledge about the executed part of a process marks the track on the terrain that the project has passed already. The simulated variants of prediction knowledge, when superimposed on the map, are essentially the suggestions of the continuations of a design process.

Knowledge workers in MIC design may be facilitated by using such an expert navigator in a variety of ways. One important use case is for making optimized project plans by exploiting the experience of the other teams in their accomplished projects. A project manager at the project planning phase is challenged by the necessity of making numerous choices of: (i) the functional blocks from the available IP libraries to be reused; (ii) a particular sequence of technological operations that best suits the type of the developed design

[7] http://www.mediawiki.org/wiki/Extension:LiquidThreads

[8] LiveNetLife is an application for contextualized interactive real-time communication between the users of a web site: http://www.livenetlife.com/

artifact and results in the best performance level; (iii) a most efficient and effective tool for a particular activity; (iv) the constellation of designers with matching skill sets in the development team. These choices can be made based on the analyses of the choices made in the successfully accomplished projects, especially those performed by experienced alpha teams.

Another use case exploits the synergy of the ACTIVated software and the ProjectNavigator simulation functionality. Once the project plan is detailed down to the level of a Work Breakdown Structure, it may be fed back to the ProjectNavigator simulator for exploring how it would be executed. The simulation results further articulated and shared within the development team using the DP Visualizer provide the proofs of the quality and reliability of the developed project plan.

3 Validation Methodology and Results

The software demonstrator described above implements both back-end and front-end functionalities that have been validated. The methodology for back-end software validation is essentially an analytical evaluation of the components and the acquired data and knowledge. It aims at checking the quality and the completeness of data as well as the robustness of software components where applicable. The validation of the front-end part is an empirical assessment by professional users of various aspects like conformance to the requirements, appropriateness of use and usability. The majority of these validation kinds are based on the use of specific questionnaires that the users have to answer after a period of software trials. These trials are organized in the settings defined specifically for each validation kind. The feedback is collected and further analyzed. The users involved in the validation trials are Cadence DP managers also possessing significant experience as the members of development teams for various types of products. Dry runs[9] have been performed leading to the refinement of the requirements to the developed software. Appropriateness and usability trials have been arranged around the four typical tasks of DP managers (upper part of Fig. 1). It has been observed in the preliminary interviews with the users that the most effective way to assess the utility of the demonstrator is to compare the way of executing and the results of the typical tasks in the following two cases: (i) before the introduction of ACTIVE technologies; and (ii) using the ACTIVated software. The trialists have been asked to use the DP Visualizer in the mentioned typical tasks and provide their subjective assessments in terms of the increase in performance and the decrease of the required ramp-up effort. The trialists filled-in the specifically developed questionnaires online based on their personal assessments of the work with the offered tools.

[9] A dry run is a validation trial for checking the conformance of a software artifact to the requirements.

For the front-end the dry run trials in three iterations as well as usability validation have been performed. The results for the final software revision indicate that the development is consistent with the plan. The software demonstrates reasonable conformance to the specified requirements. User perception of the demonstrator as the tool for their typical tasks is satisfactory in terms of the increase of performance. The trialists also believe that the combination of the offered technologies in the collaboration platform will facilitate to a substantial decrease of ramp-up efforts. A detailed discussion and analysis of the validation set-up and experiments are presented in [6].

4 Conclusions and Future Work

This paper presented the solution for design project knowledge articulation and sharing developed in the case study of ACTIVE project. Validation experiments proved that these functionalities are helpful for a design project manager in executing the typical tasks of project planning. The subjective assessments of the validation trialists allow believing that the introduction of the tool into everyday industrial practice may increase the performance and facilitate to decreasing the ramp-up effort of project managers and the members of their development teams.

Extending the sets of the source facts and mining more reliable assertions from these is one of the directions for our future work. Another direction is extending the visualization front-end with the functionality that generates other visualized representations of design process plans and executions. For example, the extension of the front-end that generates Gantt diagrams both for plans and runs may allow including project execution monitoring in the sphere of the coverage of our solution.

References

1. Crapo, A.W., Waisel, L.B., Wallace, W.A., Willemain, T.R.: Visualization and the process of modeling: a cognitive-theoretic view. In: KDD 2000: Proc. of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 218–226. ACM, New York (2000)
2. Dello, K., Nixon, L., Tolksdorf, R.: Extending the makna semantic wiki to support workflows. In: Proc. of the 3rd Semantic Wiki Workshop (2008)
3. Dengler, F., Lamparter, S., Hefke, M., Abecker, A.: Collaborative process development using semantic MediaWiki. In: Proc. of the 5th Conf. of Professional Knowledge Management, Solothurn, Switzerland (2009)
4. Drucker, P.F.: The age of discontinuity: guidelines to our changing society. Peter F. Drucker. Heinemann, London (1969)
5. Ell, B., Dengler, F., Djordjevic, D., Garzotto, F., Krötzsch, M., Siorpaes, K., Vrandecic, D., Wögler, S.: Conceptual models for enterprise knowledge final models and evaluation. Tech. Rep. D1.1.2, ACTIVE Project (2010)

6. Ermolayev, V., Jentzsch, E., Ivasiva, A., Fortuna, C., Dengler, F.: Demonstrator and validation report. Tech. Rep. D10.2.2, ACTIVE Project (2010)
7. Ermolayev, V., Jentzsch, E., Keberle, N., Sohnius, R.: Performance simulation initiative. the suite of ontologies v.2.3. reference specification. Technical Report PSI-ONTO-TR-2009-1, VCAD EMEA Cadence Design Systems, GmbH (2009)
8. Ermolayev, V., Keberle, N., Matzke, W.: An upper level ontological model for engineering design performance domain. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 98–113. Springer, Heidelberg (2008)
9. Ermolayev, V., Matzke, W.: Towards industrial strength business performance management. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 387–400. Springer, Heidelberg (2007)
10. Ermolayev, V., Matzke, W., Sohnius, R.: Engineering design performance. In: UNISCON, pp. 108–110 (2008) (Invited talk)
11. Grobelnik, M., Mladenic, D., Ferlez, J.: Probabilistic temporal process model for knowledge processes: handling a stream of linked text. In: Conf. on Data Mining and Data Warehouses (SiKDD 2009). Ljubljana, Slovenia (2009)
12. Gronau, N., Müller, C., Korf, R.: KMDL - capturing, analysing and improving Knowledge-Intensive business processes. J. Universal Computer Science 11(4), 452–472 (2005)
13. Mayer, R., Menzel, C., Painter, M., de Witte, P., Blinn, T., Perakath, B.: Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Tech. rep., Knowledge Based Systems Inc. (1995)
14. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Readings in speech recognition, pp. 267–296. Morgan Kaufmann Publishers Inc., San Francisco (1990)
15. Ryu, K., Yücesan, E.: CPM: a collaborative process modeling for cooperative manufacturers. Advanced Engineering Informatics 21(2), 231–239 (2007), Ontology of Systems and Software Engineering; Techniques to Support Collaborative Engineering Environments
16. Scheer, A.W., Jost, W.: ARIS in der Praxis. Springer, Berlin (2002)
17. Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., van de Velde, W., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press, Cambridge (1999)
18. Sohnius, R., Jentzsch, E., Matzke, W.: Holonic simulation of a design system for performance analysis. In: Mařík, V., Vyatkin, V., Colombo, A.W. (eds.) HoloMAS 2007. LNCS (LNAI), vol. 4659, pp. 447–454. Springer, Heidelberg (2007)
19. Warren, P., Kings, N., Thurlow, I., Davies, J., Bürger, T., Simperl, E., Moreno, C.R., Gomez-Perez, J., Ermolayev, V., Ghani, R., Tilly, M., Bösser, T., Imtiaz, A.: Improving knowledge worker productivity - the ACTIVE integrated approach. BT Technology Journal 26(2) (2009)
20. Woitsch, R., Karagiannis, D.: Process oriented knowledge management: A service based approach. J. Universal Computer Science 11(4), 565–588 (2005)

Part VII
Modeling and Security

High-Level Petri Nets with Object-Orientation – A High-Level Petri Net Model Allowing Multiple Inheritance and Multiple Polymorphism

Marius Brezovan, Dumitru Burdescu, Eugen Ganea, and Liana Stanescu

Abstract. This paper presents a high-level Petri net model called High-Level Petri Nets with Object-Orientation (HLPNOO), a new approach for introducing object-oriented concepts into the framework of Petri nets. An important feature of the HLP-NOO formalism is the fact that it allows distinct hierarchies for subtyping and sub-classing. We use order-sorted algebras in order to specify the notion of subtyping for object type hierarchies. Moreover we use the notions from category theory and institutions in order to allow composable Petri nets and multiple inheritance. We use encapsulated multi-methods and a multi-dispatching mechanism for messages in order to safety integrate the concepts of covariant and contravariant specialization of inherited methods and to allow the multiple polymorphism.

1 Introduction

Since last decades, numerous proposals for associating object-oriented concepts and methods with Petri nets into a single framework, which combine the expressive power of both approaches were defined and implemented. According to [2], three directions of integration may be differentiated: (a) proposals that integrate object-oriented concepts into Petri-nets, including formalisms such LOOPN [5] and OBJSA [9], (b) proposals that integrate Petri-nets into object-oriented techniques, including formalisms such CLOWN [3] and OCPN [6], and (c) proposals that mutual integrate object-oriented techniques and Petri-nets, including formalisms such CO-OPN/2 [10], COO [1], LOOPN+ [15], and OOPN [4].

Nevertheless, the situation in this field of the higher-level Petri net models is far from satisfactory. The continuous interest in Petri nets has been generated a

Marius Brezovan · Dumitru Burdescu · Eugen Ganea · Liana Stanescu
University of Craiova
e-mail: mbrezovan@software.ucv.ro, dburdescu@software.ucv.ro, eganea@software.ucv.ro, lstanescu@software.ucv.ro

multitude of dissimilar approaches containing different notions and techniques, many of the object-oriented extensions of Petri nets are made ad hoc and lack a formal theoretical background. On the basis of these motivations, we present the results of our investigation towards a suitable OO Petri net model, referred to as HLPNOO (High-Level Petri Nets with Object-Orientation), that is based on a complete and a sound integration of OO concepts into an appropriate variant of the High-Level Petri Nets standard [8] denoted by Extended High-Level Petri Nets with Objects [11].

It is widely recognized that subtyping and inheritance are mechanisms concerned with the manipulation of types and implementations, respectively [17]. Nevertheless, a majority of the modern OO programming languages and formalisms, including Petri net approach, have opted for the unification of the notions of type and implementation into a single concept. HLPNOO formalism separates the notion of a class in two distinct elements: (a) an *interface*, used to define an object type, and (b) an *implementation module*, used to specify an object type implementation.

One important debate concerns the use covariant or contravariant specialization of inherited methods in presence of polymorphism. However, as far as we know up to now there is no OO Petri net formalism where these two concepts are safety integrated. In order to integrate these concepts we use *encapsulated multi-methods* and a multi-dispatching mechanism for messages [12, 13], instead of ordinary methods and a single dispatching mechanism.

The rest of the paper is organized as follows. Section 2 presents the notions concerning object types and interfaces. Section 3 presents the notions concerning the implementation modules. Section 4 presents the formal definition of HLPNOO, and Sect. 5 concludes the paper.

2 Interfaces and Types

Throughout this paper we assume a universe U, which includes several disjoints sets, $U = SORT \cup OID \cup VAR \cup METH$, where $SORT$ is the set of all sorts, OID is the set of all identifiers associated to objects, VAR is the set of variable names, and $METH$ is the set of public method names of object types. The set VAR is assumed to be $SORT$-sorted, and the set $SORT$ is assumed to be partitioned in three disjoint sets, $SORT_D$, $SORT_{Ob}$ and $SORT_{Imp}$, corresponding to the names of (non object-oriented) data types, the names of object types, and the name of implementation modules for object types respectively. The set $METH$ is considered as a sorted set:

$$(METH_{\hat{c}w,s})_{c \in SORT_{Ob},\ w \in SORT^*,\ s \in SORT \cup \{\varepsilon\}} \tag{1}$$

For each method name $m \in METH_{\hat{c}w,s}$, $\hat{c}$ represents the reference type to the object containing the method, w is the sequence of sorts representing the input values, and s is the sort of the returned value. The function type of all functions having arguments of sorts $w = s_1 \ldots s_n$ and result of the sort s is denoted by by $w \rightarrow s$.

An *encapsulated multi-method* is an overloaded method associated with an object type, which contains several normal methods (branches) having the same name [12].

The type of a multi-method is denoted as $\{\hat{c}w_1 \to s_1, \dots, \hat{c}w_n \to s_n\}$, where $\hat{c}w_i \to s_i$ is the function type of the i^{th} branch of m.

Definition 1. Let c be an object type. An **encapsulated multi-method** of c with n branches is a pair (m,t), where $m \in \bigcap_{i=1}^{n} METH_{\hat{c}w_i,s_i}$ is the name of the multi-method, and $t = \{\hat{c}w_1 \to s_1, \dots, \hat{c}w_n \to s_n\}$ represents its type. The set of all possible encapsulated multi-methods of c is denoted by $MMETH(c)$.

In order to compare the sorts of function types we use the subtyping relation. Let $u, v \in SORT^*$, and $s,t \in SORT$. If $v \le u$ and $s \le t$, then in the case of function types we have $u \to s \le v \to t$. Subtyping relation for function types allows us to define a similar subtyping relation for multi-methods. Let $t_1 = \{\hat{c_1}w_1 \to s_1, \dots, \hat{c_1}w_m \to s_k\}$ and $t_2 = \{\hat{c_2}v_1 \to r_1, \dots, \hat{c_2}v_n \to r_n\}$ be two encapsulated multi-method types. The type t_1 is a *subtype* of t_2 if for every function type $\hat{c_2}v_j \to r_j$ from t_2 there exists a function type $\hat{c_1}w_i \to s_i$ from t_1 such that $\hat{c_1}w_i \to s_i \le \hat{c_2}v_j \to r_j$.

When defining interfaces we use the weak form of substitutability principle because interfaces are defined at syntactic level.

Definition 2. Let c be an object type. An **interface** defining c is a tuple, $intf = (c, super, meth)$, where $super \in SORT_{Ob}^*$ is a sequence of object types specifying the subtype relation associated to c, and $meth \subseteq MMETH(c)$ is a finite set of encapsulated multi-methods of c. The set of all interfaces is denoted by $INTF$.

For an interface $intf = (c, super, meth)$, with $super = s_1 \dots s_n$, the subtype relation associated to c is denoted by $\le_{intf}$ and it defines a partial order on $SORT_{Ob}^*$:

$$\le_{intf} = \{(c,s_1), \dots, (c,s_n)\}. \tag{2}$$

Let $intf_1 = (c_1, super_1, meth_1)$ and $intf_2 = (c_2, super_2, meth_2)$ be two interfaces defining the object types c_1 and c_2. The object type c_2 is a *subtype* of c_1, and it is denoted as $c_2 \le_S c_1$, if the following relations hold: (a) $(c_2, c_1) \in \le_{intf_2}$, (b) the subtype c_2 has at least all the messages of the supertype c_1, and (c) multi-methods of c_2 may have subtypes of the corresponding multi-methods of c_1 with the same name.

Additionally, some restrictions on a set of interfaces have to be imposed in order to define a coherent set of object types. Let $IT = \{intf_1, \dots, intf_n\}$ be a set of interfaces, with $intf = (c_i, super_i, meth_i)$, $i = 1, \dots, n$. The *first requirement* prevents two distinct interfaces to be associated with the same object type, that is $intf_i, intf_j \in IT$ with $i \ne j$ implies $c_i \ne c_j$. The *second requirement* prevents an object type to be defined as a subtype of itself. Let $\le_{IT}$ be the set of all subtype relations of IT. The set $\le_{IT}$ determines the subtype graph associated to IT, and the second requirement constraints this directed graph to be acyclic. A set of interfaces IT is *well-defined* if IT verifies the above two requirements.

3 Modules and Implementations

The implementation module of an object type is realized in the HLPNOO formalism by using a class of Petri nets called *Extended High-Level Petri Net with Objects* (EHLPNO) [11], which represent high level Petri nets enriched with some object orientation concepts, such as creating new objects inside transitions when they fire, and calling public methods of objects inside transitions. In HLPNOO to each method or branch from an encapsulated multi-method corresponds a subnet contained in its associated Petri net, having an input place which stores the input values of the method, and an output place where the returned values are presented.

There are two main differences between High-Level Petri Nets as presented in the ISO standard [8] and EHLPNOs: (a) the use of order-sorted algebras instead of many-sorted algebras in order to ensure the subtype polymorphism, and (b) the use of actions inside transitions in order to specify some object-oriented concepts. Moreover, in order to ensure the inheritance relation for implementation modules, the these nets are based on the more abstract concept of institutions with semi-composable signatures. An institution $\mathbf{INS} = \langle \mathbf{Sign}, \mathbf{Sen}, \mathbf{Mod}, (\models_\Sigma)_{\Sigma \in |\mathbf{Sign}|} \rangle$ has (finitely) *semi-composable signatures* if its category of signatures **Sign** is (finitely) cocomplete and the model functor **Mod** maps pushouts in the category **Sign** of signatures to pullbacks in the category **Cat** [18].

We propose the institution $\mathbf{NOS} = \langle \mathbf{Sign_{NOS}}, \mathbf{Sen_{NOS}}, \mathbf{Mod_{NOS}}, (\models_\Sigma)_{\Sigma \in |\mathbf{Sign_{NOS}}|} \rangle$ of regular order-sorted signatures with a Noetherian ordering on the sorts and signature morphisms preserving maximal elements of connected components, order-sorted algebras and theories with conditional sort constraints, which is a variant of the the well known institution **OSA** [14]. The institution **NCOS** proposed in [7] is more restrictive than our institution, requiring the connected components to have a maximal element. We omit this restriction because according to [14], each regular order-sorted signature can be extended to a coherent signature.

It can be shown that the institution **NOS** has composable signatures, and the category of algebras $\mathbf{Mod_{NOS}}(\mathbf{Sig_{NOS}})$ is finitely cocomplete. In addition, the category $\mathbf{Sign_{NOS}}$ is an inclusive category and it has pullbacks. As a consequence, the amalgamated sum of a two signatures, Σ_1 and Σ_2, over a common signature Σ_0, $\Sigma_1 +_{\Sigma_0} \Sigma_2$, can be extended to the *plain union* operator of two signatures, $\Sigma_1 + \Sigma_2 = \Sigma_1 +_{\Sigma_0} \Sigma_2$, where $\Sigma_0 = \Sigma_1 \cap \Sigma_2$. Corresponding to signatures, the amalgamated sum of two algebras, A_1 and A_2 over a common algebra A_0, can be extended to the plain union of the two models.

The second main difference between High-Level Petri Nets and EHLPNOs is the extension of annotations associated to transitions. Two types of actions are considered: (a) creating new objects inside transitions when they fire, and (b) calling public methods of objects inside transitions.

Let t be a transition and x be a variable associated to an object ob. A *method call* is a syntactical construction having one of the the two following forms:

$$\begin{gathered} x.m(a_1, \ldots, a_n) \\ b \leftarrow x.m(a_1, \ldots, a_n) \end{gathered} \tag{3}$$

where $m \in METH$ is the name of a method of the object ob, $a_1, \dots, a_n$ are expressions containing input variables of t, and b is an output variable of t. The first form is associated to a procedure call from programming languages, whereas the second form is associated to a function call. The set of all method calls is denoted by *MCALL*.

The creation of objects inside of a transition, t, can be specified by using the following syntactical form, which represents an *instruction for creating objects*:

$$x = new\ impl\ (e_1, \dots, e_n) \tag{4}$$

where x represents a variable associated to the newly created object, $e_1, \dots, e_n$ represent parameter expressions containing input variables of t, and *impl* is the name of the implementation module of the created object. The set of all object creation instructions is denoted by $CREATE$. From a syntactical point of view, a *create* method of an implementation module *impl* has a form similar to an encapsulated multi-method, $(create, t_{create})$, where t_{create} represent the type of the set of all *create* methods of *impl*. The type of the set of all function types of the *create* methods of the module *impl* is denoted by $TCreate(impl)$.

Definition 3. Let **NOS** be the above specified institution, IT a set of well-defined interfaces, $Sig = (S, \leq, O)$ a Boolean order-sorted signature from $\mathbf{Sig_{NOS}}$, and $H = (S_H, \leq, O_H)$ an order-sorted *Sig*-algebra from the model functor $\mathbf{Mod_{NOS}}$. An **Extended High-Level Petri Net with Objects** is a tuple,

$$ehlpno = (NG, Sig, V, H, Type, AN, AC, M_0), \tag{5}$$

where:

(i) the net graph NG, the type function $Type$, the net annotation AN, and the initial marking M_0 are defined as in the standard of High-Level Petri Nets;
(ii) V is an S-indexed set of variables, disjoint from O, containing at least the reserved variable $self$;
(iii) $AC : T \to CREATE \cup MCALL \cup \{undef\}$ is the action annotation:

- if $AC(t) \in CREATE$ has the form defined by Eq. 4, where $e_i \in TERM(O \cup V)_{s_i}$, $i = 1, \dots, n$, x is a variable having the sort $\hat{c_1}$, and the module *impl* implements the object type c, with $c_1 \leq c$, then $\hat{c}s_1 \dots s_n S \to \varepsilon \in TCreate(impl)$;
- if $AC(t) \in MCALL$ has the form defined by Eq. 3, where $b \in TERM(O \cup V)_s$, $a_i \in TERM(O \cup V)_{s_i}$, $i = 1, \dots, n$, and x is a variable having the sort $\hat{c}$, then there exists an interface $intf \in IT$, $intf = (c_1, super, meth)$, with $c_1 \leq c$, such that $m \in METH_{\hat{c_1}s_1 \dots s_n, s}$ and there exists an encapsulated multi-method $(m, t_m) \in meth$, and $\hat{c_1}s_1 \dots s_n \to s \in t_m$;

The set of all Extended High-Level Petri Nets with Objects is denoted by $EHLPNO$.

In the above definition $TERM(O \cup V)_s$ represents the set of terms of sort s built over the signature $Sig = (S, \leq, O)$ and variables from the set V.

In order to define the inheritance relation for implementation modules we use two requirements: (a) compositional algebraic specifications, and (b) compositional Petri nets. The former requirement is accomplished by using the framework of institutions, and the later will be accomplished by defining a cocomplete category of EHLPNO.

Definition 4. Given the institution **NOS**, a set of well-defined interfaces IT, and $ehlpno = (NG, Sig, V, H, Type, AN, AC, M_0)$ an EHLPNO, a **Categorical Extended High-Level Petri Nets with Objects** is a tuple,

$$cehlpno = (P, T, Sig, V, H, Type, Pre, Post, TC, AC, \widehat{M_0}), \tag{6}$$

where P, T, Sig, V, H, $Type$, TC, and AC are as in Def. 3, and

(a) The functions $Pre, Post : T \rightarrow (P \times TERM(O \cup V))^{\oplus}$ represent the pre- and post-mappings defining for each transition with adjacent arcs the corresponding arc inscriptions.
(b) The initial marking of $cehlpno$ is defined as $\widehat{M_0} \in (P \times (\biguplus_{s \in S} H_s))^{\oplus}$.

The set of all Categorical Extended High-Level Petri Nets with Objects is denoted by $CEHLPNO$.

In the above definition $B^{\oplus}$ denotes the free commutative monoid generated by B, having the same meaning as μB, the set of all multisets over the set B. It can be proved that there is an equivalence between the sets $EHLPNO$ and $CEHLPNO$. The length of the paper do not alow us to describe the construction of the EHLPNO-morphisms. Extended High-Level Petri Nets with Objects and EHPLPNO-morphisms form a category denoted by **EHLPNO**, and it can be proved that **EHLPNO** is finitely cocomplete, has inclusions and pullbacks. Figure 1 presents an example of amalgamated union of two nets.

We can now define the notion of implementation module. Each implementation module is associated to a single object type that it implements, but several implementation modules may implement the same interface. An *implementation module* of the interface $intf$ defining the object type c is a tuple:

$$impl = (sm, ehlpno, Create, intf, inherit) \tag{7}$$

where $sm \in SORT_{Imp}$ is the name of the module, $ehlpno$ is an Extended High-Level Petri Net with Objects, $Create = (create, t_{create})$ is the set of all $create$ methods of c, and $inherit \in SORT^{*}_{Imp}$ is a sequence of implementation module names specify implementation inheritance relation of $impl$.

Because methods in HLPNOOs are represented by subnets from EHLPNOs, the inheritance relation concerns the compositionality of EHLPNOs.

Definition 5. Let $impl_i = (sm_i, ehlpno_i, Create_i, intf_i, inherit_i)$, $i = 0, 1, \ldots, k$, be $k+1$ implementation modules.

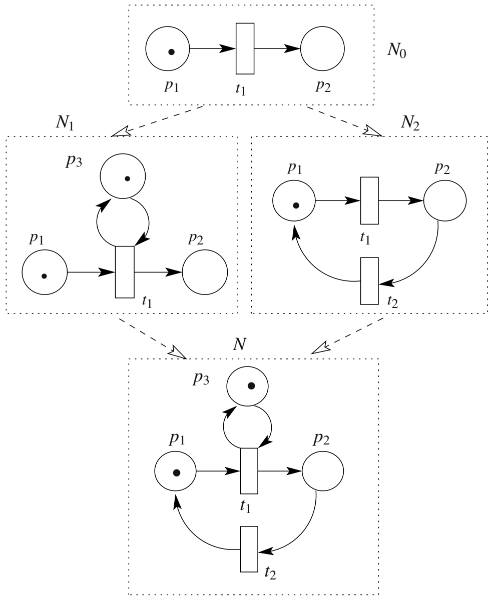


Fig. 1 Example of amalgamated union of two EHLPNOs: $N = N_1 +_{N_0} N_2$

(1)The module $impl_0$ **simple inherits** the module $impl_i$, iff $inherit_0 = intf_i$, and $ehlpno_i$ is a subnet of $ehlpno$;

(2)The module $impl$ **multiple inherits** the modules $impl_1, \dots, impl_k$, iff $inherit_0 = intf_1, \dots, intf_k$, and $ehlpno$ is the plain union of $ehlpno_1, \dots, ehlpno_k$.

For each module $impl_i$ inherited inherited by $impl_0$ it is denoted that $impl_i \leq_I impl_0$. As in the case of interfaces, two additional constraints are imposed on a set of implementation modules IM in order to define coherent implementations: (a) the first requirement prevents two distinct implementation modules to be associated with the same name, and (b) the second requirement constraints the directed graph associated to IM to be acyclic. A set of implementation modules IM is *well-defined* if IM verifies the above two requirements.

4 High-Level Petri-Nets with Object-Orientation

Definition 6. Let IT be a set of well-defined interfaces, IM a set of well-defined modules that implement interfaces from IT. An **object-oriented system** associated to IT and IM is a triple:

$$OS = (IT, IM, Oid) \tag{8}$$

where $Oid : IM \rightarrow \wp(OID)$ is a function which associates a set of object identifiers to each implementation module, such that $Oid(impl_i) \cap Oid(impl_j) = \emptyset$ if $impl_i \neq impl_j$, or $Oid(Impl_i) \subseteq Oid(impl_j)$ if $Impl_i \leq_I impl_j$.

The object identifier sets $Oid(impl_i), i = 1, \ldots, n$ are disjoint sets in the case of unrelated implementation modules, in order to prevent two objects from different implementation modules to have the same object identifier.

Definition 7. Let $OS = (IT, IM, Oid)$ be a object-oriented system as in the above definition. An **Object-Oriented High Level Petri Net** associated to OS is a triple:

$$hlpnoo = (OS, impl_0, oid_0) \tag{9}$$

where $impl_0 \in IM$, and $oid_0 \in Oid(impl_0)$ is the object identifier associated to the initial object of $hlpnoo$.

The initial implementation module $impl_0$ of an HLPNOO is important when defining the dynamic semantics of these nets. It represents the higher level of abstraction for a modelled system, and its initial object is the unique instance of $impl_0$, which exists at the beginning of the dynamic system evolution.

We illustrate the way of using EHLPNOs when defining an HLPNOO. As an example consider the well-known maze game. The maze contains one human and four enemies. Five classes are considered: $CGame$, $CMaze$, $CElement$, $CHuman$ and $CEnemy$, where $CHuman$ and $CEnemy$ inherit the class $CElement$. The functionality of the class $CGame$ allows to human and enemies to move alternatively. The structure of the corresponding EHLPNO is presented in Fig. 2, where both transitions t_1 and t_2 call the public method *move* of the classes $CHuman$ and $CEnemy$. respectively. The input place and the output places of the method *move* are denoted by #*move* and *move*# respectively.

The class *CMaze* contains the sequence of the cells of the maze, each cell being represented by the type of the element its contains (a human, an enemy, or empty). *CMaze* has two public methods: *update* which updates the contents of the cells, and *getPos* which returns the set of the cells. The class *CMaze* has a very simple functionality and it contains only a single transition t corresponding to the update operation. The class $CElement$ is a base class for $CHuman$ and $CEnemy$, and contains a public method, *move*, that allows to realize a movement. The structure of the corresponding EHLPNO is presented in Fig. 3. The transition t calls the method

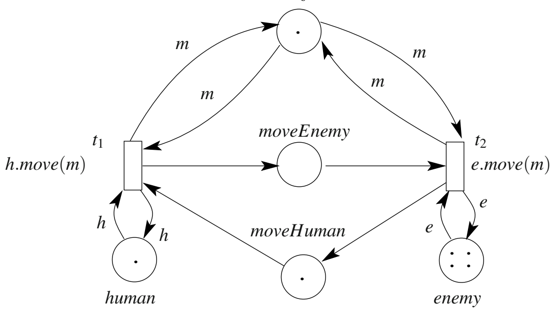


Fig. 2 EHLPNO of the class Game

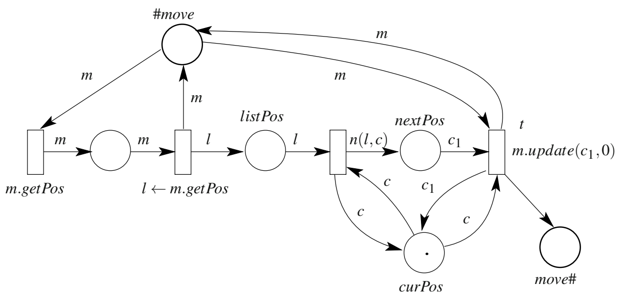


Fig. 3 EHLPNO of the class Element

update of the class *CMaze*. The first argument of the method *update* specifies the current cell to be updated, while the second argument specifies the type of its content. The place *listPos* contains the set of the cells of the maze, and $n(l,c)$ is a function defined in the algebra associated to *CMaze*, which implements the strategy of the next movement. The first argument of n specifies the list of the cells, while the second argument specifies the current cell.

5 Conclusions

In this paper, we have proposed a new class of Petri nets, called High-Level Petri Nets with Object-Orientation which encapsulate the object-oriented methodology into the Petri net formalism. The HLPNOO formalism is based on the ISO standard of High-Level Petri Net and it can be easily integrated in projects concerning concurrent and object-oriented systems.

The main contributions of our proposal can be sumarized as follows: (a) Unlike other approaches, HLPNOO use two distinct types of hierarchies: a hierarchy for object types and subtyping, and a different hierarchy for inheritance. (b) In order to safety integrate the concepts of covariant and contravariant specialization of inherited methods and to allow the multiple polymorphism we use encapsulated multimethods and a multi-dispatching mechanism for messages. (c) We use a variant of the institution of OSA and a categorical form of EHLPNOs in order to allow Petri nets to be composable and thus to allow multiple inheritance for HLPNOO.

Future work will be carried out in two main directions: (a) to study thoroughly the connection between rule-based systems and HLPNOO and to develop a method for implementing dynamic semantics using an inference engine for rule-based systems, and (b) to develop a graphical and text-based language for HLPNOO and to build a tool supporting this language.

Acknowledgements. This work was supported by CNCSIS – UEFISCSU, project number PNII – IDEI code/2008.

References

1. Bastide, R., Sibertin-Blanc, R., Palanque, P.: Cooperative objects: A concurrent, Petri-net based, object-oriented language. In: Proc. of the IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 286–292 (1993)
2. Bastide, R.: Approaches in unifying Petri Nets and the Object-Oriented Approach. In: Proc. of the 1st Workshop on Object-Oriented Programming and Models of Concurrency, Torino, Italy (1995)
3. Battiston, E., Chizzoni, A., Cindio, F.: Inheritance and concurrency in CLOW. In: Proc. of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Torino, Italy (1995)
4. Ceska, M., Janousek, V.: Object Orientation in Petri Nets. In: Proc. of the 22nd Conference of the ASU, Clermont-Ferrand, France, pp. 69–80 (1997)
5. Lakos, C.A., Keen, C.D.: LOOPN - Language for Object-Oriented Petri Nets. In: Proc. of the SCS Multiconference on Object-Oriented Simulation, Anaheim, USA, pp. 22–30 (1991)
6. Maier, C., Moldt, D.: Object Coloured Petri Nets - A formal technique for object oriented modelling. In: Farwer, B., Moldt, D., Stehr, M.O. (eds.) Petri Nets in System Engineering, Modelling, Verification and Validation, pp. 11–19. University of Hamburg (1997)
7. Mossakowski, T.: Representations, hierarchies and graphs of institutions. PhD thesis, Universitat Bremen (1996)
8. ISO/IEC 15909-1, Software and system engineering. High-level Petri nets. Part 1: Concepts, definitions and graphical notation (2004)
9. Battiston, E., Cindio, F.D., Mauri, G.: OBJSA nets: A Class of High-level Nets having Objects as Domains. In: Rozenberg, G. (ed.) APN 1988. LNCS, vol. 340, pp. 20–43. Springer, Heidelberg (1988)
10. Biberstein, O., Buchs, D., Guelfi, N.: Object-oriented nets with algebraic specifications: The CO-OPN/2 formalism. In: Agha, G.A., De Cindio, F., Rozenberg, G. (eds.) APN 2001. LNCS, vol. 2001, pp. 70–127. Springer, Heidelberg (2001)
11. Brezovan, M.: A formal definition of High Level Petri Nets with Objects. Annals of the University of Craiova. Series: Electrical Engineering 26, 45–54 (2002)
12. Bruce, K., Cardelli, L., Castagna, G.: The Hopkins Objects Group, Leavens G, Pierce B, On Binary Methods. Theor. Pract. Obj. Syst. 1(3), 221–242 (1996)
13. Castagna, G.: Covariance and Contravariance: Conflict without a Cause. ACM T Progr. Lang. Sys. 17(3), 431–447 (1995)
14. Goguen, J., Meseguer, J.: Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partialoperations. Theor. Comput. Sci. 105(2), 216–263 (1992)
15. Lakos, C.A.: Object Oriented Modelling with Object Petri Nets. In: Agha, G.A., De Cindio, F., Rozenberg, G. (eds.) APN 2001. LNCS, vol. 2001, pp. 1–37. Springer, Heidelberg (2001)
16. Reisig, W.: Petri nets and algebraic specifications. Theor. Comput. Sci. 80, 1–34 (1991)
17. Taivalsaari, A.: On the Notion of Inheritance. ACM Comput. Surv. 28(3), 438–479 (1996)
18. Ehrig, H., Mahr, B.: Fundamentals of Algebraic Specifications I. In: Equations and Initial Semantics. EATCS Monograph in Theoretical Computer Science, vol. 6, Springer, Heidelberg (1985)

Mobile Agents with Timers, and Their Implementation

Gabriel Ciobanu and Călin Juravle

Abstract. Implementing code mobility is a difficult task, especially when migration is under some time constraints. In this paper we present a language for mobile agents starting from an architecture and a formalism with timed interactions and explicit locations. The language supports the specification of a distributed system, i.e. agents and their physical distribution, and allows a timed migration in a distributed environment.

1 Introduction

In this paper we present a high-level language for mobile agents starting from a formalism with timed interactions and explicit locations. We use an advanced software technology for the development of mobile agents in distributed environments. Using the specification of the language, appropriate parser and editor are generated. We use the existing technology to generate the code, and to obtain the agents derived from the formal specification; these agents are executed over a general software platform which can be used to implement various systems based on mobility. The implementation corresponds rigorously to the formal semantics.

We employ a technology called openArchitectureWare which supports development of domain-specific languages. By using *Xtext* which is a part of openArchitectureWare, we describe a language for which a parser together with an editor are generated. The language supports the specification of mobile agents in a distributed system, i.e. agents and their physical distribution. We generate code by using the template language *Xpand*. The implementation corresponds rigorously to the semantics of the language. We provide a general way of implementing complex distributed systems that involve code

Gabriel Ciobanu · Călin Juravle
Institute of Computer Science, Romanian Academy, Iaşi
and "A.I.Cuza" University of Iaşi, Romania
e-mail: `gabriel@iit.tuiasi.ro, gabriel@info.uaic.ro`

mobility and timed communications. The defined software platform can be used for various systems with mobility, being able to handle low-level details such as network or location management.

It is widely recognized that migration and interaction (communication) represent challenging aspects in multi-agent systems. In general, agents provide a way to describe complex systems by entities capable of acting with a certain degree of autonomy in order to accomplish tasks. The fundamental idea of an architecture for mobile agents is that it identifies components and their interconnections. We use an open architecture which allows new interconnections between locations and agents. Separating the migration mechanism, this can be used by various systems involving mobility. The architecture facilitates the implementation of mobile agents, being able to handle low-level details such as network or location management.

There are various approaches to code mobility. A simpler approach is when a piece of code is sent out to a computing host where the piece of code is executed. There exist several mobile agent platforms (based on Java mobility); we mention here Aglets [9], JADE [2], and D'Agents [7]. TACOMA [8] and Mobile-C [3] are mobile agent platforms supporting C/C++ mobile agent code. A Mobile Agent-based GriD Architecture (MAGDA) was designed to support mobile agents for grids [1]. In more advanced systems, a mobile agent can move from a host to another and continue its code execution. We present such an advanced system. The advantage of such a system over the simpler ones is that it reduces the network load between the hosts. Another advantage is that once a mobile agent starts its movement, the host can be disconnected from the network (e.g. to minimize the energy consumption).

2 Mobile Agents Architecture

Implementing code mobility is a difficult task, especially when time is considered. Looking to the multi-agent systems from the standpoint of architecture, we define an architecture and implement a software platform starting from this architecture. The interconnections among agents is described in operational terms. the specification of architecture and interconnections in terms of higher-level abstractions allows more flexibility and a richer family of implementations.

The architecture for mobile agents is called *MAA*; it allows the link between the specification of various formalisms for mobility and the practical aspects which deal with mobile code. MAA is designed to abstract the concepts used in agent systems, and to handle low-level details such as network or location management. This architecture has an associated GUI which facilitates the interaction with the formal framework. Using such a GUI, it is easy to access the majority of framework functionalities without any coding: it is possible to start or stop the system, change the active formalism (the upper layer), compile, load and execute specific formal specifications and interact with other architecture.

The architectures serves as an abstract framework for implementation of models for mobility, dealing with the common part of such formalisms: names, locations, agents, migration, fresh name generation, etc. It provides a default mechanism for mobility which makes possible to migrate an agent by migrating its code and data. Moreover, it also provides the architecture of an engine for simulation of the formal evolution of an agent. Since the framework handles communications with other machines, it can create and initialize a distributed environment from a specification, making it a useful tool for distributed experiments. Usually the specification of a distributed environment is provided by a concrete implementation. The framework is based on an extensible architecture so that the majority of components can be customized according to needs. It is implemented in Java language, the main reason being the infrastructure offered by Java for working with mobile code and dynamic classes. The software platform is implemented by extending structures from the architecture and adapting them to its specific features.

From a functional viewpoint, the architecture can be represented by two layers and the dependencies between and inside these layers. The lower layer is given by the MobileCalculi framework and the upper layer is given by the formal specification of the agents. We focus here on the migration module. This module creates the needed infrastructure for agent migration. It also provides a default migration mechanism based on bytecode migration. This module abstracts the migration objects by providing an interface which must be implemented by all the entities which want to migrate.

An agent describes a software abstraction, a concept similar to OOP terms such as methods, functions, and objects. Actually we connect these concepts, and represent an agent by an object which contains a method with its actions. The main entities of our system for mobility are agents (*MAgent*), locations (*Location*) and names (*Name*). The agent runs at a specific location, in a private thread. The location acts as an execution context for agents. It keeps a list with resources, such as communication channels, which can be used by agents. All entities (including resources) are referred to by name, so the name concept is also embodied in a class. An agent may have, besides its body, a formal representation which will be used, if present, for formal simulation. In order to make possible agent migration all mentioned entities are serializable and marked as *MigratingCode*. The migration module works with objects of type *MigratingCode*, which is an interface that must be implemented by all entities which want to migrate. In particular, it is implemented by *MAgent*. This maintains a decoupled architecture and makes it possible to easily change the migration mechanism. The migration is realized by *MigrationCodeMarshaller* and *MigrationCodeUnmarshaller*. The marshaller is used on the local machine when the agent has to be sent, and the unmarshaller is used on the remote machine when the agent is received. Both objects, *MigrationCodeMarshaller* and *MigrationCodeUnmarshaller*, are created by a

MigrationCodeFactory. In order to customize the migration mechanism, the only step is to extends this class with the desired features.

Migration uses the infrastructures provided by the *MobileCalculi* framework. The framework implements a mobility mechanism in which the programmers retain the program counter and manage the point from which the execution of agents is restarted at destination. The semantic of migration timers is implemented by using a distributed protocol. The local timer *lt* represents the time dedicated to internal and local actions executed before migration. It is decremented, and after this local timer reaches 0, the migration procedure is initiated and the migration timer starts to be decremented. The migration timer *mt* represents the maximum time dedicated to migration; it is implemented as a distributed protocol. After the agent arrives at its destination, a *receive message* is sent back. If the message is receive before the migration timer becomes 0 the agent is remove from its initial location and another message, a *confirmation message* is sent to destination; otherwise the agent activates its safety process at the current location. At destination the agent restarts only after receiving the *confirmation message*. The default behaviour if this message is not received is to remove the agent (at destination). A successful migration can be visualized in Figure 1.

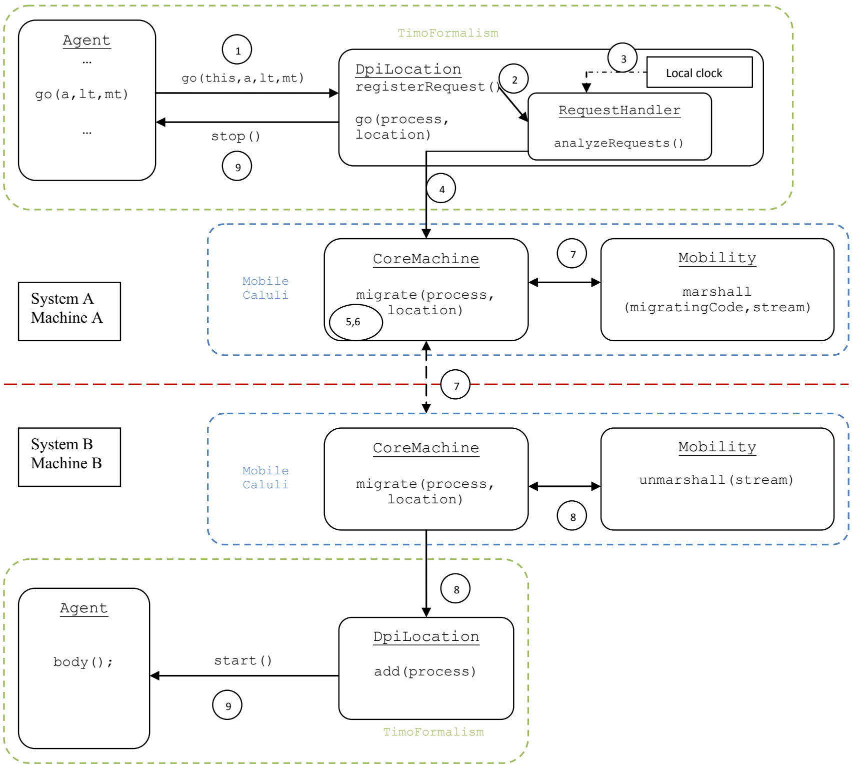


Fig. 1 Migration of an agent

The temporal aspects are implemented with the help of a *virtual clock*. The clock is local to each location, and so has a predefined frequency. At each tick it triggers an event, and the subscribers take the appropriate actions; each location analyzes at every tick the requests it has received.

We implement a software package called *MATools*, and emphasize the use of openArchitectureWare technology (part of the Eclipse Modeling project) as a technology which supports development of Domain Specific Languages for mobility. openArchitectureWare consists of a set of tools for developing model-driven software by using meta modelling, creation of domain-specific languages, code generation and model transformation. In particular, we use *Xtext* and the code generation in order to obtain the objects derived from a formal specification; these objects are executed using *MATools*. *MATools* can be used to implement various systems based on mobility. Besides a migration mechanism, it offers generic implementations of common concepts needed to implement mobile systems. We describe the software package *MATools*. *Xtext* is used in implementing a language for mobile agents with timers, and *Xpand* to generate code. The integration of the generated code and their execution using *MATools* is exemplified.

3 Language for Mobile Agents with Timers

In order to help writing agents, we should have a language to intermediate between the high-level language and the low-level Java code. We briefly present the syntax of the language and its operational semantics. The language is close to a process algebra specification in which one can formally model distributed systems with explicit locations, migration and temporal constraints. It could be related to π-calculus family [10], more exactly to the timed distributed π-calculus [6]. It features a simple syntax, dropping the type aspects of timed distributed π-calculus and focusing on timed interaction and migration. Time is local, and it is modeled by timers which are associated with basic actions. The result is that the time of an action is no longer indefinite. Moreover, if an action does not happen in a predefined time, then the agent continues with a safety alternative. The time constraints associated with input and output restricts the period in which these capabilities could be applied. After the predefined deadline is reached, another continuation for the agent behaviour is chosen. For instance, if an agent $c^{\Delta 3}\,\mathbf{?}\,(u)\,\mathtt{then}\,P\,\mathtt{else}\,Q$ does not receive a value on channel c in the next 3 units of local time, it will behave according to agent Q.

The time constraint associated with migration is composed from two timers. The first one is the *local time*, and it represents the time dedicated to internal and local actions which are executed before migration. The second one is the *migration time*, and it represents the maximum time dedicated to migration. For instance, an agent $m[\![\mathbf{go}^{4\Delta 10}\,k\,\mathtt{then}\,P\,\mathtt{else}\,Q\,]\!]$ executes local actions for 4 units of time (at the current location m), and then it migrates in maximum 10 units of time to location k where it behaves as P. Note that since v in $\mathbf{go}^{lt\Delta mt}\, v\, \mathtt{then}\, P\, \mathtt{else}\, Q$ is a variable, migration supports a flexible movement of agents between locations.

3.1 Syntax and Semantics

It is assumed that *Chan* is a set of channels, *Loc* is a set of locations, *Var* is a set of location variables and *Ident* is a finite set of agent identifiers (each identifier $I \in Ident$ has a fixed arity $m_I \geq 0$).

The high-level language for mobile agents with timers is defined by

$$\begin{aligned} P, Q ::= \;& 0 \;\mid\; a^{\Delta t}\,\mathbf{!}\,\langle v\rangle\, \mathtt{then}\, P\, \mathtt{else}\, Q \;\mid\; a^{\Delta t}\,\mathbf{?}\,(u)\, \mathtt{then}\, P\, \mathtt{else}\, Q \;\mid \\ & \mid\; \mathbf{go}^{lt\Delta mt}\, v\, \mathtt{then}\, P\, \mathtt{else}\, Q \;\mid \\ & \mid\; I(v_1, \ldots, v_{m_I}) \;\mid\; P \,|\, Q \;\mid\; \#P \\ M, N ::= \;& k[\![P]\!] \;\mid\; M \,|\, N \end{aligned}$$

In the above description it is assumed that $a \in Chan$; $t, lt, mt \in \mathbb{N}$,; $v, v_1, \ldots, v_{m_I} \in Loc \cup Var$; $k \in Loc$ and $u \in Var$. Moreover, each agent identifier $I \in Ident$ has a unique definition of form $I(u_1, \ldots, u_{m_I}) = P_I$ where $u_i \neq u_j$ (for $i \neq j$) are variable acting here as parameters.

Output agent $a^{\Delta t}\,\mathbf{!}\,\langle v\rangle\, \mathtt{then}\, P\, \mathtt{else}\, Q$ attempts to send v over channel a for t units of time. If the communication takes place then it continues as P, otherwise it continues as Q. Input agent $a^{\Delta t}\,\mathbf{?}\,\langle u\rangle\, \mathtt{then}\, P\, \mathtt{else}\, Q$ is waiting to receive a value in u; if the communication is successful then it continues as P, otherwise it continues as Q. Agent $\mathbf{go}^{lt\Delta mt}\, v\, \mathtt{then}\, P\, \mathtt{else}\, Q$ is used to describe the movement. The agent waits lt units of time which represents the *local time* dedicated to local work, then it moves to location v in mt units of time (mt stands for migration time). If the move is accomplished within the specified time, then the agent behaves as P (at v), otherwise it continues as Q at current location. Agents are further constructed from the basic defined agents or by using the parallel composition $P \,|\, Q$. A located agent $k[\![P]\!]$ is an agent running at location k. The symbol $\#$ from $\#P$ is a purely technical notation used in the formalization of structural operational semantics. Intuitively, it says that the agent has finished its action and it is temporally waiting for the next tick of the clock.

Operational Semantics is given by the rules presented in Table 1.

Looking to the labels of the transitions, there are two kinds of transition rules: $M \xrightarrow{\beta} N$ and $M \xrightarrow{\surd} N$. The first one corresponds to the execution of an action β, while the second one represents a timing tick. The action β can be either $k : l$ or $k : a(l)$, where k is the location where the action takes place, l is either the location where the agent goes, or the location transmitted along the channel a. In rule TIME, $N \not\rightarrow$ denotes that no other rule can be applied. ϕ is the time-passing function. Each top-level expression $I(l_1, \ldots, l_{m_I})$ is replaced by the corresponding definition $\{l_1/u_1, \ldots, l_{m_I}/u_{m_I}\}P_I$. Each top-level expression of the form $a^{\Delta 0} \ldots \mathtt{then}\, P\, \mathtt{else}\, Q$ or $\mathbf{go}^{0\Delta 0} \ldots \mathtt{then}\, P\, \mathtt{else}\, Q$ is

Table 1 Operational Semantics

GO:	$k[\![\mathbf{go}^{0\Delta mt}\, l\, \mathtt{then}\, P\, \mathtt{else}\, Q\,]\!] \xrightarrow{k:l} l[\![\#P]\!]$
COM:	$l[\![a^{\Delta t}\,\mathbf{!}\,\langle l\rangle\, \mathtt{then}\, P\, \mathtt{else}\, Q \;\mid\; a^{\Delta t'}\,\mathbf{?}\,(u)\, \mathtt{then}\, P'\, \mathtt{else}\, Q'\,]\!] \xrightarrow{k:a(l)} l[\![\#P \mid \#\{l/u\}P']\!]$
PAR:	$\dfrac{N \xrightarrow{\beta} N'}{N \mid M \xrightarrow{\beta} N' \mid M}$
STRUC:	$\dfrac{N \equiv N' \quad N \xrightarrow{\beta} M \quad M \equiv M'}{N' \xrightarrow{\beta} M'}$
TIME:	$\dfrac{N \not\rightarrow}{N \xrightarrow{\surd} \phi(N)}$

replaced by $\#Q$. For the top-level communication expressions with $\Delta t > 0$, t is decreased by 1. Each top-level expression of the form $\mathbf{go}^{lt\Delta mt}\, \mathtt{then}\, P\, \mathtt{else}\, Q$ is replaced by $\mathbf{go}^{lt'\Delta(mt')}\, \mathtt{then}\, P\, \mathtt{else}\; Q$ where $lt' = \max\{0, lt-1\}$ and $mt' = mt$ if $lt > 0$ or by $mt' = max\{0, mt-1\}$ if $lt = 0$. All occurrences of the special symbol # are deleted. A top-level expression is not containing a symbol #. Note that only after the lt timer reaches 0, the agent migrates to the destination.

4 OpenArchitectureWare

OpenArchitectureWare is part of the Eclipse Modelling Framework (EMF), representing a new development technology of Domain-Specific Languages (DSL). OpenArchitectureWare consists of a set of tools able of generating code starting from EMF models.

Xtext is a textual DSL development framework. It offers the possibility of describing a DSL using an extended BNF notation. *Xtext* creates a parser, a metamodel and a specific Eclipse text editor. The syntax defined by using the extended BNF notation is not only used as input for the parser generator, but it is also used to compute a metamodel for the domain-specific language. Finally, the code generation starting from this textual model is using a template language called *Xpand*.

The openArchitectureWare framework contains a template language called *Xpand* that is used to control the output generation. A template file consists of IMPORT statements, EXTENSION statements and one or more DEFINE blocks (called definitions). The definitions represent the central concept of *Xpand*; they can be seen as special methods of the metaclass, and their body can contain a sequence of statements. An EXPAND statement is similar to a subroutine call, "expanding" another definition in a separate variable context, inserting its output at the current location and going on with the next statement. Expression statements support processing of the information provided by the instantiated metamodel. *Xpand* provides powerful expressions for selection, aggregation, and navigation. It performs the code generation, i.e. run the templates. The *Xpand* templates are evaluated on the metamodels by the generator engine. Running the generator produces the code. Whenever a template is executed, additional template can be used to generate additional code (for example, meta-information for frameworks). More details can be found on the *openArchitectureWare* website www.openarchitectureware.org.

4.1 Code Generation

We describe the code generation process starting from the language for mobile agents with timers. Using the Xtext framework, we create a domain specific language for mobile agents. Based on this language, we develop an Eclipse text editor and a code generator. With the help of Xtext we describe the *intermedLang* grammar in an enhanced BNF notation. From that intermediate language, using Xtext Editor, we generate a full Eclipse Text Editor featuring auto-completion and error signalling. The Xtext generator produces the code for MATools framework which consist of Java classes together with the system descriptor (i.e., locations addresses and agent distribution). The final result is a single file containing: locations, agent distribution and agent file paths. This file can be passed to *MATools* platform and in this way we can load and run it.

intermedLang is developed to link the high-level language used to describe the mobile agent with the low-level framework *MATools*. It has an intuitive syntax close to both agent language and the conventional programming; *intermedLang* is created to facilitate the development of mobile agents for execution in distributed environments. *intermedLang* uses only a limited set of Java types and instructions, focusing on the primitives used by the agent language. The allowed types are *Integer*, *Boolean* and *String* (together with location and parameterized channels). The instructions which can be used, besides the language primitives, are *if then else* and *while*. The *intermedLang* syntax is inspired by the language for mobile agents, so it is easier to implement agents having a formal specification. It provides a way to specify an entire system, including real addresses of machines, preferred communication ports and agents distribution map. Since it keeps features from conventional programing style, it is easier to share data (such as a channel name) between agents through global variables. Moreover, *intermedLang* provides a simple way of writing (private) channels, known only by some agents. The channel declaration must be qualified with "private" keyword and the generator will ensure that the channel name is unique (it makes use of MATools mechanism for generating fresh names). *intermedLang* simulates successfully strong mobility; however, the programmer still has to take care to ensure that the agent starts from the desired execution point. Another advantage is that the language is well typed, any type inconsistency resulting in a compiling error. For example, if an agent is coded to receive a *String* over channel a, but the channel is dedicated to a location variable, then an error is triggered. Most of these errors appear at the writing time, by using the editor plug-in. Apart from this, the development of *intermedLang* programs is easier with the syntax highlighting and auto-completion features.

A high-level structure from the agent description becomes a low-level implementation by defining two functions, $agent2prog$ and $prog2impl$, which translate first an agent description into a program, and then the intermediate program into a MATools implementation. We work with two functions, namely $agent2prog : Agent \rightarrow intermedLang$ and $prog2impl : intermedLang \rightarrow MATools$. Since the agents are built structurally, it is sufficient to define these functions for basic syntactic structures. The agents can be encoded directly in Java (MATools) without using the intermediate language (program). This can easily be proved by composing the functions $agent2prog$ and $prog2impl$. All these claims express that, using using openArchitectureWare, we have a sound code generation for mobility starting from the high-level language for mobile agents. We conclude with the statement that every mobile agent can be implemented to be executed on the platform *MATools*, and its execution reflects faithfully the operational semantics of the language.

The use of an intermediate program in *intermedLang* is given by installing its plug-in into the Eclipse distribution. This enables *intermedLang* projects to be created. After writing the agent, the code is generated by the oAW workflow file. Execution of this file creates a file containing the system description. This file is passed to MATools framework which loads and runs it.

5 Conclusion

Agent systems are used to model real world systems with concurrency or parallel processing. Mobile agents paradigm becomes increasingly popular for network-centric programming. Usually the design of mobile agent systems requires addressing several issues at the system level such as the provision of code mobility, object naming, portability and scalability. Agent programming requires suitable languages and programming models that can support code mobility. This means software platforms that provide fundamental primitives for the creation, migration, and management of the agents. The challenges of programming languages for mobile code are not new [11].

In this paper we define a high-level language for mobile agents with timers starting from a general theoretical approach of describing distributed systems coming from the process algebra [6] simplified then in [5]. Several theoretical results of these papers [5, 6] are applicable to the language and the platform presented in this paper where we emphasize the use of the new software technology in implementing complex systems involving code mobility and timed intercommunication. First we define a software architecture, and present the syntax and semantics of the language for mobile agents. Then we derive an implementation from using an advanced technology for creating a platform for mobile agents with timers. A related platform derived from [5] is presented in [4]. Here we use a new software technology called openArchitectureWare. Using the Xtext framework of openArchitectureWare, we create a domain specific language for the high-level language of mobile agents. Using Xtext Editor, we generate a text editor featuring auto-completion and error signalling. The language syntax is used as input for the parser generator; it is also used to compute a metamodel for the domain specific language. Finally, the code generation starting from this textual description is using the *Xpand* framework of openArchitectureWare.

To summarize, we provide a general way of implementing complex systems involving code mobility and timed communications, and emphasize the use of an advanced software technology called openArchitectureWare. Further work includes more powerful languages and tools for more subtle situations.

Acknowledgements. Many thanks to the anonymous referees for their helpful comments. This work was partially supported by CNCSIS' project IDEI 402/2007 and Dinu Patriciu Foundation's project "Orizonturi deschise.

References

1. Aversa, R., Martino, B.D., Mazzocca, N., Venticinque, S.: MAGDA: A Mobile Agent Based Grid Architecture. Journal of Grid Computing 4, 395–415 (2006)
2. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: JADE: A Software Framework for Developing Multi-Agent Applications. Information and Software Technology 50, 10–21 (2008)
3. Chen, B., Cheng, H.H., Palen, J.: Mobile-C: A Mobile Agent Platform for Mobile C/C++ Agents. Software Practice and Experience 36, 1711–1733 (2006)
4. Ciobanu, G., Juravle, C.: A Software Platform for Timed Mobility and Timed Interaction. In: Lee, D., Lopes, A., Poetzsch-Heffter, A. (eds.) FMOODS 2009. LNCS, vol. 5522, pp. 106–121. Springer, Heidelberg (2009)
5. Ciobanu, G., Koutny, M.: Modelling and Verification of Timed Interaction and Migration. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE 2008. LNCS, vol. 4961, pp. 215–229. Springer, Heidelberg (2008)
6. Ciobanu, G., Prisacariu, C.: Timers for Distributed Systems. Electronic Notes in Theoretical Computer Science 164, 81–99 (2006)
7. Gray, R.S., Cybenko, G., Kotz, D., Peterson, R.A., Rus, D.: D'Agents: Applications and Performance of a Mobile Agent System. Software Practice and Experience 32, 543–573 (2002)

8. Johnansen, D., Lauvset, K., van Renesse, R., Schneider, F.B., Sudmann, N.P., Jacobsen, K.: A TACOMA Retrospective. Software Practice and Experience 32, 605–619 (2002)
9. Lange, D.B., Oshima, M.: Programming and Deploying Java Mobile Agents with Aglets. Addison-Wesley, Reading (1998)
10. Milner, R.: Communicating and Mobile Systems: the π-calculus. Cambridge University Press, Cambridge (1991)
11. Thorn, T.: Programming Languages for Mobile Code. ACM Computing Surveys 29, 213–239 (1997)

Improving the Security of CHAP Protocol by Quantum Cryptography

Mohamed Elboukhari, Mostafa Azizi, and Abdelmalek Azizi

Abstract. Quantum Cryptography or Quantum key distribution (QKD) is new method that allows the secure distribution of a bit string, used as key in cryptographic protocols. Extensive researches have been undertaken on Quantum Cryptography since it was noted that quantum computers could break public key cryptosystems based on number theory. Quantum Cryptography offers unconditionally secure communication based on laws of quantum physics. Actually, there is a strong need of research to use this technology in the existing communication networks. We explore in this paper, the possibility of exploiting QKD in the authentication process for local area networks (LANs). Precisely, we have elaborated a scheme to integrate the new technique of QKD in the CHAP protocol (Challenge Handshake Authentication Protocol) to enhance the security of the authentication service.

1 Introduction

Unlike classical cryptography that relies on the conjectured difficulty of computing certain functions, the security of Quantum Cryptography or Quantum Key Distribution (QKD) [1] is guaranteed by the postulate of quantum physics. QKD enables two distant parties (say Alice and Bob) to derive a secret key that has guaranteed privacy due to the use of quantum physics. The secret key thus provides perfect security when it is used in One Time Pad cryptosystem [2]. QKD generated more interest after the first practical demonstration over 30 cm of free space employing polarization coding [3].

Mohamed Elboukhari
FSO, University Mohamed Ist,Oujda, Morocco
e-mail: `elboukharimohamed@gmail.com`

Mostafa Azizi
ESTO, University Mohamed Ist, Oujda, Morocco
e-mail: `azizi.mos@gmail.com`

Abdelmalek Azizi
Academy Hassan II of Sciences and Technology, Morocco
e-mail: `abdelmalekazizi@yahoo.fr`

Now, a hot issue of research is to integrate QKD technology in the existing protocols to achieve highest degree of security [4]-[5]. The LANs networks give many interests relating to the use of Quantum Cryptography. Firstly, the limited coverage area of LANs networks is adequate to implement QKD because Quantum Cryptography is actually experimented over tens of kilometers. Secondly, the wireless LANs is usually used to provide Internet access; this kind of application is critical from a network security point of view because users can realize e-commerce or banking transactions via the Internet and these applications need a very strong security that Quantum Cryptography can offer.

In this spirit, we treat in this paper the task of integrating QKD in LANs networks; we have elaborated a technique to integrate QKD in the CHAP protocol. Using the BB84 QKD protocol [6], we defined an extended CHAP protocol (QKD-CHAP) which improves the security of CHAP protocol as it has introduced in RFC1994 [7].

The remainder of the paper is organized as follows. In section 2, we introduce our new scheme of CHAP protocol (QKD-CHAP protocol) integrating the service of QKD. We conclude the paper in section 3 by giving the main results.

2 Integration of Quantum Cryptography in the CHAP Protocol: QKD-CHAP Protocol

2.1 Why Integrating QKD in the CHAP Protocol?

The first weakness of the CHAP protocol is the sharing of the same secret (S) between the Peer and the Authenticator for a long time. In this case, the attacker (Eve) who knows the function $HASH$, the number H and the value of $HASH(H,S)$ can by applying the brute attack deduct the value of the secret S by using different value of its test S_{eve} in the equation $HASH(H,S) = HASH(H,S_{eve})$ until he finds $S = S_{eve}$. We suppose here that the attacker has an advanced technology to do so, as possessing a quantum computer which is much faster than any of our current classical computers.

The Peer and the Authenticator, to overcome this type of attack, can use the same secret S for a short time only and thus they exchange different values of the secret S. But this strategy demands to exchange at each time the secret with a guaranteed security which is impracticable. To generate such secrets, we use some protocols of key distribution as Diffie-Hellman protocol [2] but unfortunately the security of such protocols is computational, which means it depends on mathematical assumptions and then it is not an unconditional security.

In our new scheme QKD-CHAP protocol, we do not need to exchange the secret S several times between the Peer and the Authenticator but the secret is shared only one time. Next, in the value of $HASH(H,S)$ implicated in CHAP protocol, we replace the secret S by the key generated by the mechanism of QKD noted QS (Quantum Secret) which is exchanged with an unconditional security due to the application of the law of quantum physics. More interestingly, to render the task of the attacker more complicated, the value of the secret QS is modified at every session of the CHAP protocol. Finally, our extension of CHAP protocol (QKD-CHAP protocol) provides a mutual authentication.

2.2 The QKD-CHAP Protocol Packet Format

In order to configure the mechanism of QKD between the Peer and the Authenticator, it is necessary to exchange some information between them. In our scheme QKD-CHAP protocol, we propose a specific CHAP packet format to carry QKD parameters as illustrated in Fig. 1.

Code		Identifier	Length
Key-Length			
TTL	T	Data ...	

Fig. 1 The QKD-CHAP Packet format

Plus the standard filed of CHAP packet format, the QKD-CHAP packet format contains new fields of Key-Length, TTL, and T. Their description is as follows:

Key-length (four octets): This field shows the length of the key generated by the execution of the BB84 protocol. We choose the length to be so huge if we plan to use the One Time Pad because in such case the key size must be equal to the data which will be encrypted [2].

TTL filed (one octet minus one bit): This field shows the number of messages or an amount of time (in seconds) where a key can be used in the encryption process. Once the max of message is reached or the time expires, a new session of QKD starts.

T field (one bit): This field specifies if the TTL field is shown as a number of message or an amount of time. If its value is ”1”, the TTL is valued as a number of messages and if its value is ”0”, then the TTL is presented as an amount of time. We note that the Length field is the length of the all filed presented in the QKD-CHAP packet.

2.3 Description of the QKD-CHAP Protocol

Fig. 2 summarizes how different messages are exchanged between the Peer and the Authenticator during our extended version of CHAP protocol, QKD-CHAP protocol:

1) After having received the number H, both the Peer and the Authenticator agree on values of field of Key-Length, TTL and T by sending each other the QKD-CHAP packet which is encapsulated in the PPP Data Link Layer frame [8].

2) The Peer which plays the role of Alice begins the first phase of BB84 protocol by sending photons one by one to the Authenticator (Bob) over the quantum channel.

3) The Peer (Alice) and the Authenticator, after the emission of the photons is completed, begin the second phase of BB84 protocol. As in this phase all the messages are public and as BB84 protocol is vulnerable to the attack "man in the middle" [6], we must face such risk. For that we propose that all the messages exchanged during this phase are protected by a hash function $HASH$ as [9] and with the initial secret S to ensure the integrity and the authenticity of the messages.

If Alice sends to Bob for example the information $INFO$, she sends to him the couple $(INFO, HASH(INFO,S))$. Receiving this two information, Bob computes its own value of $HASH(INFO,S)$ using the value received $INFO$ and the shared secret S ($HASH$ is a public data) and then he checks the authenticity (and also the integrity) of the information $INFO$. Here, we note even the mutual authentication is completed our scheme will improve the security by varying continuously the secret S used in the function $HASH$ at each new session of QKD-CHAP protocol. Once the public phase of BB84 is finished, Alice and Bob share the possession of a secret key $K = \{0,1\}^N$ if there is no attack on quantum channel, we give it the name of quantum secret $K = QS$.

4) Alice calculates $HASH(H,QS)$ and sends it to Bob. Receiving this value, Bob computes its own value $HASH(H,QS)$ and compares the two values and then decides in the end if the authentication succeeds or not.

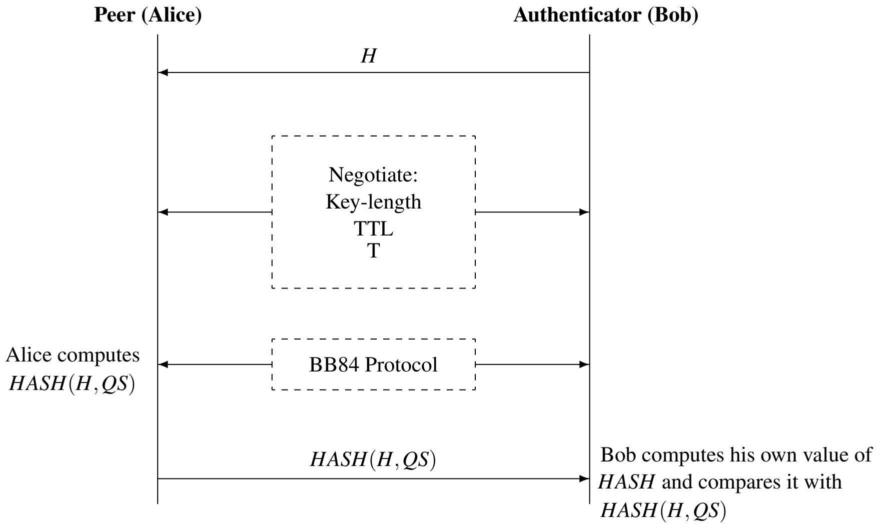

Peer (Alice)
Authenticator (Bob)
H
Negotiate:
Key-length
TTL
T
Alice computes HASH(H,QS)
BB84 Protocol
HASH(H,QS)
Bob computes his own value of HASH and compares it with HASH(H,QS)

Fig. 2 The Message flow for the QKD-CHAP protocol

In the next regular interval, Alice and Bob use the common secret QS as a new secret instead of the secret S and repeat the stages 1), 2), 3) and 4). By doing this QS is exploited to assure the authenticity and integrity of the public messages of BB84 protocol and another secret key QS_2 generated by the current execution of the BB84 protocol is employed in the value of $HASH(H_2, QS_2)$ sent by Alice, here H_2 is the value emitted by Bob (Authenticator) during this next session of QKD-CHAP protocol and in this case , any information $INFO$ sent during the BB84 public phase are protected by the couple $(INFO, HASH(INFO, SQ))$.This scenario modifies the secret S implicated in the value of the hash function $HASH$ at each session of QKD-CHAP protocol.

3 Conclusion

We have presented in our article a scheme QKD-CHAP protocol which integrates Quantum cryptography to enhance the security of the authentication process. we have precisely integrated the BB84 protocol in the standardized solution of CHAP protocol. Our novel extension of CHAP (QKD-CHAP) provides several advantages:
1. The secret is modified in each new session of QKD-CHAP protocol, this complicate the task of an attacker.
2. The secret shared between the Authenticator and the Peer is not exchanged over the network even it varies at each session of QKD-CHAP protocol.
3. The security of the secret used in the hash function is unconditional due to the laws of quantum physics.
4. Our extension provides a mutual authentication during the exchanging of the public messages of BB84 protocol.

References

1. Gisin, N., et al.: Quantum cryptography. Rev. Mod. Phys. 74, 145–195 (2002)
2. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C, 2nd edn. Wiley, Chichester (1995)
3. Bennett, C.H., et al.: Experimental quantum cryptography. J. Cryptol. 5, 3–28 (1992)
4. Nguyen, T.M.T., Sfaxi, M.A., Ghernaouti-Hélie, S.: 802.11 i Encryption Key Distribution Using Quantum Cryptography. JOURNAL OF NETWORKS 1(5), 9 (2006)
5. Elboukhari, M., Azizi, M., Azizi, A.: Integration of Quantum Key Distribution in the TLS Protocol. IJCSNS 9(12), 21–28 (2009), http://paper.ijcsns.org/07_book/200912/20091204.pdf (Cited June 07, 2010)
6. Bennett, C.H., Brassard, G.: Quantum cryptography: Public key distribution and coin tossing. In: Proc. IEEE Int. Conf. Computers, Systems and Signal Processing, New York, Bangalore, India, pp. 175–179 (1984)
7. Simpson, W.: PPP Challenge Handshake Authentication Protocol (CHAP), RFC 1994 (August 1996)
8. Simpson, W.: The Point-to-Point Protocol (PPP), RFC1661 (July 1994)
9. Rivest, R.: The MD5 message-Digest Algorithm, RFC 1321 (April 1992)

Simulation of Botnets: Agent-Based Approach

Igor Kotenko, Alexey Konovalov, and Andrey Shorov

Abstract. The paper considers an approach intended to investigate botnets and botnet defence using agent-based simulation. We explore various botnet attacks and counteraction against them on the example of defence against Distribute Denial of Service (DDoS) attacks. We represent botnet and defence components as agent teams. Agents are supposed to collect information from various network sources, operate different situational knowledge, and react to actions of other agents. The paper describes the common simulation framework, agent-based simulation environment, and the results of experiments aimed to investigate botnets and DDoS defence mechanisms.

Keywords: modelling and simulation of intelligent distributed systems, multi-agent systems, network attacks and defence, botnets.

1 Introduction

Today, botnets, as networks of malicious programs, occupy the leading positions in the lists of current threats to the Internet. The distinctive features of modern botnets are a wide variety of possible targets of attacks, including the theft of personal or any other confidential data, spam, DDoS attacks, use of infected computers in their own purposes, compromise of legitimate users, cyber blackmail, fraud of rating tracking systems. Functioning of botnets is characterized by simultaneous actions of a great number of software agents in the interests of malefactors.

The paper considers an approach for investigation of botnets and botnet defence mechanisms. The approach is based on the agent-based simulation of cyber-attacks and cyber-defence mechanisms which combines discrete-event simulation,

Igor Kotenko · Alexey Konovalov · Andrey Shorov
Laboratory of Computer Security Problems, St. Petersburg Institute for Informatics and Automation 39, 14 Liniya, St.-Petersburg, 199178, Russia
e-mail: {ivkote,konovalov,ashorov}@comsec.spb.ru

multi-agent approach and packet-level simulation of network protocols. Initially this approach was suggested for network attack and defence simulation in [9]. In the present paper the various methods of botnet attacks and counteraction against botnets are explored by representing attack and defence components as agent teams. The main goal of our research is to develop the usable common framework and agent-based simulation environment for analysis of botnets and botnet defence mechanisms. The paper is structured as follows. Second section describes the relevant papers and the main features of the proposed approach. Third section outlines the architecture and current implementation of agent-based simulation environment. Forth section presents the configuration of simulation environment and the examples of experiments conducted. Conclusion outlines the main results and future work directions.

2 Related Work and the Approach Suggested

Current research on botnets and botnet defence can be considered mainly in two categories botnet detection/response techniques and botnet measurement [1, 10, 17]. Botnet detection can be implemented, for example by detection via bot cooperative behaviour [8], detection by signatures of botnet communication process [2], and detection and response to attacks [13, 14, 16]. Measurement papers allow understanding the botnet phenomenon and the characteristics of specific types of botnets [4, 6]. The most dangerous classes of attacks, which are the basic attack means at use of botnets, are DDoS attacks [13]. Adequate victim protection against Botnet DDoS to constrain attack traffic can only be achieved by cooperation of different distributed components [14]. There are a lot of architectures for distributed cooperative defence mechanisms, e.g. Server Roaming, Market-based Service Quality Differentiation (MbSQD), Transport-aware IP router architecture (tIP), Secure Overlay Services (SOS), ACC pushback, COSSACK [16], Perimeter-based DDoS defence, DefCOM [14], Gateway-based.

There was developed the variety of frameworks and architectures for multi-agent modelling and simulation of distributed complex systems, e.g. shared plans theory [7], joint intentions theory [3], hybrid approach [18], there were implemented various software multi-agent environments [11, 12]. To provide the optimal interaction of heterogeneous components the approaches based on belief-desire-intention (BDI), distributed constraint optimization (DCOP), distributed POMDPs, and auctions or game-theoretic [18] are emphasized. The main task of these approaches is to reach some high level goal.

In the paper, the botnet life cycle and the process of botnet containment by defence mechanisms are simulated. Similarly to botnet structure, the structure of defence mechanisms is implemented by the subnet of defence components (agents).

In this paper the models of botnets and botnet defence are specified in the form of counteraction between the two classes of teams: the attack team and defence team. Each team represents a subset of computer network nodes, identified as agents, and having a common collective goal. Attack team includes agents belonging to the botnet and implementing actions aimed at providing the vital activity of botnets. Similarly, the defence team is made up of agents, performing the defence functions and having a collective goal to oppose against botnet operation.

The *common model of botnet and defence mechanisms*, which we are trying to implement for simulation, is represented as a tuple which consists of a set of network nodes, a set of connections between network nodes, a scenario component, and a component describing the observer (for evaluation of effectiveness of botnet and defence mechanisms).

Scenario component includes the scenarios of botnet functioning, the scenarios of botnet containment and attack counteraction, as well as scenarios of network legitimate activity. Each scenario contains the following elements: scenario goal; algorithm (rules) for goal achievement; nodes involved in the scenario. Each scenario can be iteratively detailed. In this case, each intermediate scenario, which is a part of this scenario, becomes the object of subsequent decomposition (for example, the goal of "spreading" may consist of sub-goals "scan" and "infection").

3 Multi-agent Framework and Environment

To implement the multi-agent modelling and simulation, we intended to develop a multi-level simulation environment which differs from the well-known *agent-oriented simulation tools* (e.g. CORMAS, Repast, Swarm, MadKit, MASON, NetLogo, etc.), first of all, by the use of simulation tools that allow to adequately simulate the network protocols and security processes. This environment is a software package that includes a discrete event simulator, implemented by low-level language, as well as a number of components that realize the components of higher levels. The architecture of simulation environment consists from the following four main components.

Simulation Framework is a discrete event simulation system. It provides tools for modelling chronologically ordered sequences of discrete events. The possibilities of basic model data I/O, as well as basic features on processing the results of experiments are provided.

Internet Simulation Framework is a set of modules which allow simulating nodes and protocols of the Internet. It contains the modules that form realistic network topologies, the models of network applications with behaviour close to the behaviour of real network applications, as well as the models of transport, network and link layer protocols. Each protocol is implemented as an independent module. Internet Simulation Framework also contains modules for automatic construction of standard networks based on the set of defined parameters and their automatic configuration. In current version this component uses the library ReaSE [5].

Representation of network elements as intelligent agents is realized by *Agent-based Framework*. This component is a library of modules that specify intelligent agents and common scripts of their behaviour, implemented in the form of models of services and applications embedded in the models of network nodes. The component also contains the models of application layer protocols, which provide communication between agents and interaction of agents with application models.

Subject Domain Library is a library, which serves to simulate the processes of the subject area. It includes modules that realize different botnet and botnet defence scenarios and complement the functionality of IP-node, including filter tables, packet analyzers, models of legitimate users, etc.

The proposed architecture has been implemented by using several different components: the simulator OMNeT++ [15], the libraries INET Framework and ReaSE, as well as our own software components [9].

4 Simulation Environment Configuration and Experiments

The *network topology* is simulated on two levels of detail. On the first level, the network topology on the autonomous system level (AS) is simulated. In this paper we describe the experiments in which a network, consisting of 5-10 autonomous systems (AS-level topology), is simulated. Each AS can consist of hundreds or thousands of hosts. Connections of transit AS are implemented through the communication channel with the bandwidth dr = 10000 Mbit/seconds and a delay d=1 microsecond. Connections of other AS are implemented with dr = 5000 Mbit/seconds and d = 1 microsecond. On the second level of simulation, for each AS the internal topology (Router-level topology) is simulated. The equipment of nodes has the types of “router” and “host’. The equipment of type “router” has only one functional role – “router”. The equipment of type “host” can be web server, web client, mail server, multimedia content server, “command centre”, “vulnerable service”, “master”, IP-filter.

Attack teams include the following types of nodes: “master”, “command centre”, “target”, “zombies”. Node “master”, by sending different commands, sets goals for the botnet and controls the behaviour of the network at the highest level. Node “command centre” carries out the delivery of commands received from the “master” to nodes “zombies”. Nodes “zombie”, receiving the commands from the “command centre”, immediately carry out actions under the orders of the “master. To generate a *legitimate traffic*, “server” nodes are determined. These nodes, in response to a request from “clients” generate the traffic statistically similar to traffic of standard web server.

Defence teams are represented by the following common classes of agents: information processing (“sampler”); attack detection (“detector”); filtering (“filter”); investigation (“investigator”); rate limitation (“limiter”). Samplers collect and process network data for anomaly and misuse detection. Detector coordinates the team, correlates data from samplers, and detects attacks. Filters are responsible for traffic filtering using the rules provided by detector. Investigator tries to defeat attack agents. Limiter is intended to implement cooperative defence. Its local goal is to limit the traffic according to the team goal. It lowers the traffic to the attack target and allows other agents to counteract the attack more efficiently.

To perform the experiments we have realized several *scenarios of botnet functioning (including scenarios of propagation, management and attack realization), botnet containment and attack counteraction, and network legitimate activity*.

One of the examples of implemented *scenarios of attack realization* is an attack "UDP Flood", directed to some node (subnet), IP-address of which is specified in the attack start command. We implemented several *scenarios of botnet containment and attack counteraction*: without cooperation; DefCOM-based [14]; COSSACK-based [16] and full cooperation. In defence scenario without cooperation only one defence agent team is used. In other defence scenarios we use several cooperating defence agent teams which protect different segments of computer network.

The investigation of attack and defence scenarios has been done on the basis of analysis of two *main classes of parameters*: the amount of incoming attack traffic before and after the filter of team which network is the attack victim; false positive and false negative rates of attack detection.

The experiments implemented demonstrated that full cooperation shows the best results on blocking the attack traffic. It uses several defence teams with cooperation on the level of filters and samplers.

5 Conclusion

This paper proposed an approach for simulation of botnets and botnet defence in the Internet. Botnets and botnet defence are examined by interaction of different agents teams. Environment for the agent-oriented simulation was developed on the basis of OMNeT++ INET Framework. This software simulation environment has been used to investigate various cooperative distributed defence mechanisms. The conducted experiments showed the availability of proposed approach for simulation of botnets and botnet defence. The experiments demonstrated that the use of cooperation of several defence teams leads to the essential increase of defence effectiveness.

Future work is related to comprehensive analysis of cooperation effectiveness of various attack and defence teams and inter-team interaction, the implementation of adaptation and self-learning defence to protect against manipulation by attackers, the expansion of attack and defence library, and the investigation of new defence mechanisms. The important part of future research is providing numerous experiments to study the effectiveness of prospective defence mechanisms against botnets.

Acknowledgments. This research is partly funded by the EU under SecFutur and Massif projects, the grant of the Russian Foundation of Basic Research (Project No.10-01-00826) and Program of fundamental research of the Department for Nanotechnologies and Informational Technologies of the Russian Academy of Sciences (Contract No.3.2).

References

1. Bailey, M., Cooke, E., Jahanian, F., Xu, Y., Karir, M.: A Survey of Botnet Technology and Defenses. In: The 2009 Cybersecurity Applications & Technology Conference for Homeland Security (2009)
2. Binkley, J.R., Singh, S.: An algorithm for anomaly-based botnet detection. In: SRUTI 2006, San Jose, CA (2006)
3. Cohen, P., Levesque, H.J.: Óeamwork, Nous, No. 35 (1991)
4. Dagon, D., Gu, G., Lee, C.P., Lee, W.: A taxonomy of botnet structures. In: ACSAC 2007, Florida, USA (2007)
5. Gamer, T., Scharf, M.: Realistic Simulation Environments for IP-based Networks. In: The First International Workshop on OMNeT++, Marseille, France (2008)
6. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B., Dagon, D.: Peer-to-Peer Botnets: Overview and case study. In: HotBots 2007, Cambridge, MA (2007)
7. Grosz, B., Kraus, S.: Collaborative Plans for Complex Group Actions. Artificial Intelligence 86(2) (1996)
8. Gu, G., Perdisci, R., Zhang, J., Lee, W.: BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In: Security 2008, San Jose, CA (2008)
9. Kotenko, I., Ulanov, A.: Agent Teams in Cyberspace: Security Guards in the Global Internet. In: The International Conference on CYBERWORLDS (2006)
10. Liu, J., Xiao, Y., Ghaboosi, K., Deng, H., Zhang, J.: Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures. EURASIP Journal on Wireless Communications and Networking 2009 (2009)
11. Macal, C.M., North, M.J.: Tutorial on Agent-based Modeling and Simulation. In: The 2005 Winter Simulation Conference (2005)
12. Marietto, M., David, N., Sichman, J.S., Coelho, H.: Requirements Analysis of Agent-Based Simulation Platforms: State of the Art and New Prospects. In: Sichman, J.S., Bousquet, F., Davidsson, P. (eds.) MABS 2002. LNCS (LNAI), vol. 2581, pp. 125–141. Springer, Heidelberg (2002)
13. Mirkovic, J., Dietrich, S., Dittrich, D., Reiher, P.: Internet Denial of Service: Attack and Defense Mechanisms. Prentice Hall PTR, Englewood Cliffs (2004)
14. Mirkovic, J., Robinson, M., Reiher, P., Oikonomou, G.: Distributed Defense Against DDOS Attacks. University of Delaware CIS Department Technical Report (2005)
15. OMNeT++ Community Site (2010), http://www.omnetpp.org/
16. Papadopoulos, C., Lindell, R., Mehringer, I., Hussain, A., Govindan, R.: Cossack: Coordinated suppression of simultaneous attacks. DISCEX III (2003)
17. Strayer, W.T., Lapsely, D., Walsh, R., Livadas, C.: Botnet Detection Based on Network Behavior. In: Advances in Information Security. Botnet Detection, vol. 36 (2008)
18. Tambe, M., Bowring, E., Jung, H., et al.: Conflicts in teamwork: Hybrids to the rescue. In: AAMAS 2005 (2005)

Part VIII
Parallel Computational Intelligence

Parallel Artificial Bee Colony Algorithm Approaches for Protein Structure Prediction Using the 3DHP-SC Model

César Manuel Vargas Benítez and Heitor Silvério Lopes

Abstract. This paper reports the use of the Artificial Bee Colony algorithm (ABC) for protein structure prediction using the three-dimensional hydrophobic-polar model with side-chains (3DHP-SC). Two parallel approaches for the ABC were implemented: a master-slave and a hybrid-hierarchical. Experiments were done for tuning the parameters of the ABC, as well as to adjust the load balance in a cluster-based computing environment. The performance of the parallel models was compared with a sequential version for 4 benchmark instances. Results showed that the parallel models achieved a good level of efficiency and, thanks to the co-evolution effect, the hybrid-hierarchical approach improves the quality of solutions found.

Keywords: Swarm Intelligence, Parallel Computing, Artificial Bee Colony, Protein Folding, 3DHP-Side Chain.

1 Introduction

The specific biological function of a protein depends on its three-dimensional shape, which, in turn, is a function of its primary structure (linear sequence of amino acids). Failure to fold into the intended three-dimensional shape usually leads to proteins with different properties that can become inactive or even harmful to the organism. Therefore, better understanding the protein folding process can result in important medical advancements and development of new drugs. Only a small amount of such proteins have its three-dimensional structure known. This fact is due to the cost and difficulty in unveiling the structure of proteins, from the biochemical point of view. This is an important motivation for developing computational methods for predicting the structure of these proteins. Computer science has an important role here,

César Manuel Vargas Benítez · Heitor Silvério Lopes
Federal University of Technology Paraná (UTFPR), Curitiba, PR, Brazil
e-mail: `cesarvargasb@gmail.com, hslopes@utfpr.edu.br`

proposing models for studying the Protein Structure Prediction (PSP) problem. The simplest computational model for the PSP problem is known as Hydrophobic-Polar (HP) model, both in two (2D-HP) and three (3D-HP) dimensions [4]. Although simple, the computational approach for searching a solution for the PSP using the HP models was proved to be *NP*-complete. This fact has motivated the development of many evolutionary computation approaches for dealing with the problem [8].

Currently, there are a variety of algorithms inspired by the bee foraging behavior found in literature, such as: Bee System [12], Honey Bee Algorithm [10], and the Artificial Bee Colony Algorithm (ABC) [6], among others. The ABC is a population-based optimization algorithm based on the foraging behavior of honeybee swarms. Currently, the ABC is one of the most widely used bee algorithm for problem solving. Some successful applications found in the literature using the ABC include, for instance, multi-objective problems [11], and template matching in digital images [3].

To the best of our knowledge, the current work is the first to apply parallel ABC approaches to the PSP using the 3DHP-SC model. This work compares the performance of the standard ABC sequential algorithm with two parallel models, namely: Master-Slave ABC (MS–ABC) and Hybrid Hierarchical ABC (HH–ABC). Experiments were done using four protein sequences. Therefore, one can evaluate the benefits of the parallelized approaches for the ABC algorithm in comparison with the regular sequential version.

2 Artificial Bee Colony Algorithm

Social insects, such as ants and bees, spend most of their life foraging for food. Honey bee colonies have a decentralized system to collect the food and can adjust the searching pattern precisely so that to enhance the collection of nectar.

The ABC algorithm [6] works with a swarm of n solutions x (food sources) of dimension d that are modified by artificial bees. The bees aim at discovering places of food sources v (locations in the search space) with high amount of nectar (good fitness). In the ABC algorithm, a colony of bees is divided into three groups: employed bees (forager bees) that exploit the neighborhood of their food sources selecting a random solution to be perturbed; onlooker bees (observer bees) that are placed on the food sources by using a probability based selection process; and scouts bees that randomly fly in the search space without guidance. As the nectar amount of a food source increases, the probability value P_i with which the food source is preferred by onlookers also increases. If the nectar amount of a new source is higher than that of the previous one in their memory, they update the new position and forget the previous one. If a solution is not improved by a predetermined number of trials controlled by the parameter $limit$, then the food source is abandoned by the corresponding employed bee and it becomes a scout bee. Each search cycle consists of moving the employed and onlooker bees onto the food sources and calculating their nectar amounts; and determining the scout bees and directing them onto possible food sources. The ABC pseudo-code is shown in Algorithm 1. The ABC algorithm attempts to balance exploration and exploitation by combining local search methods, carried out by employed and onlooker bees, with global search methods, managed by scout bees.

Two remarkable features of such swarm-based systems are self-organization and decentralized control that leads to an emergent behavior. Emergent behavior is a property that emerges through interactions among system components (bees) and it is not possible to be achieved by any of the components of the system by itself. Interactions between bees occurs through a waggle dance performed inside the hive. It is used to recruit other bees to exploit a food source. The intensity of the waggle dance performed by a bee is proportional to the quality of the food source. Hence, best food sources lead to a more intense waggle dance, and this can reinforce the exploitation of the best food locations.

3 The 3D-HP Side-Chain Model (3D-HP-SC)

The Hydrophobic-Polar (HP) model proposed by Dill [4] divides the 20 standard amino acids into two classes, according to their affinity to water: Hydrophilic (or Polar) and Hydrophobic. When a protein is folded in its native conformation, most hydrophobic amino acids tend to group themselves in the inner part of the protein, in such a way to get protected from the solvent by the polar amino acids that are positioned preferably outwards. Therefore, a hydrophobic core is usually formed,

Algorithm 1 Pseudo-code of Artificial Bee Colony Algorithm (ABC).

```
 1: Parameters: $n$, $limit$
 2: Objective function $f(x)$,  $x = [x_1, x_2, ..., x_d]^T$
 3: Initialize the food positions randomly $x_i$  $i = 1, 2, ..., n$
 4: ***Evaluate fitness*** $f(x_i)$ ***of the Bees***
 5: **while** stop condition not met **do**
 6:     Employed phase:
 7:         Produce new solutions with $k$, $j$ and $\phi$ at random
 8:             $v_{ij} = x_{ij} + \phi_{ij} \cdot (x_{ij} - x_{kj})$  $k \in \{1, 2, ..., n\}, j \in \{1, 2, ..., d\}, \phi \in [0,\ 1]$
 9:         ***Evaluate solutions***
10:         Apply greedy selection process for the employed bees
11:     Onlooker phase:
12:         Calculate probability values for the solutions $x_i$
13:             $P_i = \frac{f_i}{\sum_{j=i}^{n} f_j}$
14:         Produce new solutions from $x_i$ selected using $P_i$
15:         ***Evaluate solutions***
16:         Apply greedy selection for the onlookers
17:     Scout phase:
18:         Find abandoned solution: If limit exceeds, replace it with a new random solution
19:     Memorize the best solution achieved so far
20: **end while**
21: Postprocess results and visualization
```

especially in globular proteins. In this model, the folding or conformation of a protein is represented in a lattice, usually square (for the 2D-HP) or cubic (for the 3D-HP). Both 2D-HP and 3D-HP models have been frequently explored in the recent literature [8].

Since the expressiveness of the HP models is very poor, from the biological point of view, a further improvement is to include a bead to represent the side-chain (SC) of the amino acids [7]. Therefore, a protein is modeled by a common backbone and a side-chain, either Hydrophobic (H) or Polar (P).

To compute the energy of a given conformation using this model, [7] proposed an equation that takes into account the spatial position of the side-chain and the backbone, as shown in Equation 1.

$$H = \varepsilon_{bb} \sum_{i=1,j>i+1}^{N} \delta_{r_{ij}^{bb}} + \varepsilon_{bs} \sum_{i=1,j\neq i}^{N} \delta_{r_{ij}^{bs}} + \varepsilon_{ss} \sum_{i=1,j>i}^{N} \delta_{r_{ij}^{ss}} \tag{1}$$

In this equation, ε_{bb}, ε_{bs} and ε_{ss} are the weights of the energy for each type of interaction: backbone/backbone (BB-BB), backbone/side-chain (BB-SC) and side-chain/side-chain (SC-SC) In a chain of n residues, the distance (in the three-dimensional space) between the i^{th} and j^{th} residue interacting with each other is represented by r_{ij}^{**}. For the sake of simplification, in this work we used unity distance between residues ($r_{ij}^{**} = 1$). Therefore, δ is an operator that returns 1 when the distance between the i^{th} and j^{th} side-chain is the unity, or 0 otherwise. During the folding process, interactions between amino acids take place and the free energy of the conformation tends to decrease, conversely, the conformation tends to converges to its native state, in accordance with the Anfinsen's thermodynamic hypothesis [1]. In this work we consider the symmetric of H to turn the problem to a maximization.

It is known that the number of hydrophobic contacts (HnC) is inversely proportional to the free-energy of a given conformation. Therefore, any algorithmic procedure for the folding that maximizes the HnC will, conversely, take the molecule to the smallest possible free-energy state.

4 Solution Encoding and Fitness Function

The encoding of the candidate solutions (Bees) should be carefully implemented because it can have a strong influence not only in the size of the search space, but also in the hardness of the problem. To model the PSP, the solution represents the spatial position of the amino acids chain in a lattice, using internal coordinates [8]. In this coordinate system, given conformation is represented by a set of movements of one amino acid relative to its predecessor in the chain. Therefore, for a protein with n amino acids, a folding encoded in an artificial bee will have $n-1$ elements. As mentioned before, in the 3DHP-Side Chain model, the amino acids of the protein are represented by a backbone (BB) and a side-chain (SC), either hydrophobic (H) or polar (P). In the three-dimensional space there are five possible relative movements for the backbone (**L**eft, **F**ront, **R**ight, **D**own, **U**p), and other five for the side-chain, relative to the backbone (**l**eft, **f**ront, **r**ight, **d**own, **u**p). The combination of possible movements for backbone and side-chain gives a set of 25 possibilities. Instead of the traditional binary alphabet, a set of 25 numbers and letters was used to encode the chromosome. More details about the encoding was reported in a previous work [2].

To evaluate a possible solution encoded in a bee, the chromosome of relative coordinates must be previously decoded into a set Cartesian coordinates representing backbones and respective side-chains. The fitness function used in this work was first proposed by [9], and later adapted to the 3DHP-SC by [2]. This function has three terms (as shown in Equation 2). The first one is relative to the free-energy of the conformation H (see Equation 1), the number of collisions NC (number of points in the three-dimensional lattice that is occupied by more than one element) and the penalty weight (*PenaltyValue*) The following terms represent the gyration radius of the hydrophobic and hydrophilic side-chains, respectively. Radius of gyration is a measure of compactness of a set of points (in this case, the side-chains of the amino acids in the lattice). This is done in such a way to favor conformations in which hydrophobic side-chains are compacted within the core, and polar side-chains are pushed outwards of the conformation. Equations 3, 4, and 5 show how it is computed.

$$fitness = [H - (NC \cdot PenaltyValue)] \cdot RadiusG_H \cdot RadiusG_P \tag{2}$$

$$RG_{aa} = \sqrt{\frac{\sum_{i=1}^{N_{aa}} [(x_i - \overline{X})^2 + (y_i - \overline{Y})^2 + (z_i - \overline{Z})^2]}{N_{aa}}} \tag{3}$$

$$RadiusG_H = maxRG_H - RG_H \tag{4}$$

$$RadiusG_P = \begin{cases} 1 & \text{if } (RG_P - RG_H \geq 0) \\ \frac{1}{1-(RG_P - RG_H)} & \text{else} \end{cases} \tag{5}$$

5 Parallel Implementations of the ABC Algorithm

The first parallel approach was the Master-Slave (MS–ABC). It is a global single-population system where a master process distributes the processing load into several slave processes, each one running in a different processor of a cluster-based processing environment. The master is responsible for initializing the food positions randomly, generating new solutions in the algorithm phases (Employed, Onlooker and Scout – see Algorithm 1), applying a greedy selection procedure, and distributing bees (solutions) to slaves. Slaves, in turn, are responsible for computing the fitness function of received bees. The computation of the fitness function is very intensive, thus justifying the need to distribute computing. In Algorithm 1 the bold and italic instructions are where the parallelization takes place. Figure 1(a) illustrates the MS–ABC model. As shown, the MS–ABC model has exactly the same functionalities of the sequential version of the ABC.

The second parallel approach, the Hybrid Hierarchical (HH–ABC) was implemented has two levels. In the upper level, there are multiple-population coarse-grained islands (such as a multi-hive model). In the lower level, there are global single-population master-slaves (such as the MS–ABC). These two levels can be seen as an hierarchy of hives. This combination aims at taking advantage of the benefits of both models in a single approach. At the lower level there is a master process that distributes the computational effort into several slaves. In the upper level each population (master and corresponding slaves) is seen as an island that works independently. A migration policy defines the sporadic migrations that occur between islands. It has four parameters: Migration gap (number of generations between successive migrations), Migration rate (number of bees that will migrate at each migration event), Selection/Substitution criteria for migrants, and topology of connectivity between islands. Figure 1(b) illustrates the HH–ABC model with four hives at the upper level and *n* slaves per hive at the lower level.

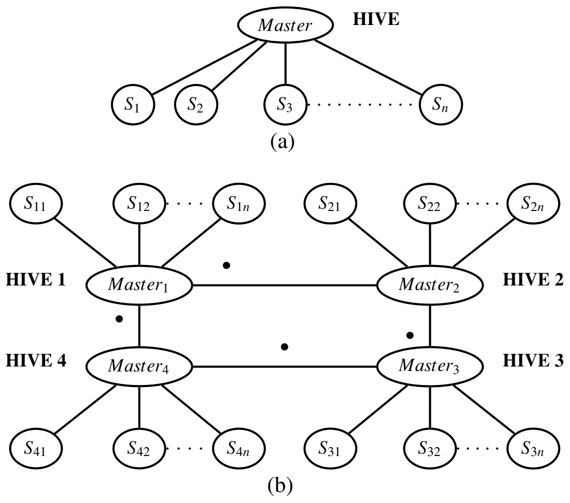


Fig. 1 MS–ABC model (a) and HH–ABC model (b)

6 Computational Experiments and Results

The experiments reported in this work were run in a cluster of networked computers with 124 processing cores, running Linux. The software was developed in ANSI-C programming language, using the Message Passing Interface (MPI) MPICH2 package for the communication between processes[1].

Similarly to other evolutionary computation algorithms, the ABC also has parameters to adjust and there is no specific procedure for doing this for a given problem.

[1] Available at: http://www.mcs.anl.gov/research/projects/mpich2/

In this work, the following parameters of the ABC algorithm were adjusted using a benchmark sequence: Colony size (*Colonysize*), Maximum Cycle Number (*MCN*), and the percentage of Employed (n_e) and Onlookers Bees (n_o). Three values for *Colonysize* and *MCN* were tested, respectively $\{250, 500, 1000\}$ and $\{6000, 3000, 1500\}$, thus keeping constant the number of function evaluations (done by employed and onlooker bees). For each combination, three pairs of values for n_e and n_o were tested, as follows: $\{75\%, 25\%\}$, $\{50\%, 50\%\}$, and $\{25\%, 75\%\}$, resulting in 9 sets of parameters. For each set, a total of 100 independent runs was done, and the average of the best fitness found in the runs was used as criterion of quality. The best performing set of parameters was: *Colonysize*=250, *MCN*=6000, n_e=50% and n_o=50%. These values were fixed for the remaining experiments.

In a cluster-based parallel processing environment, the time needed to transmit control messages and data between processes through the network is significantly high, when compared with the processors' speed. Hence, it is necessary to establish a suitable balance between the processing load of each processor and the amount of communication between processes. After several preliminary experiments, we adjusted *Colonysize* and the number of slaves (for each master-slave) in such a way to establish the best trade-off between communication and processing.

6.1 Migration and Coevolution in the HH–ABC Model

At the upper level, the HH–ABC model uses a topology with 4 islands connected by a unidirectional ring. This number of islands was set due two facts: availability of hardware (allocating one processing core per process) and the expectance of observing the coevolution effect as soon as possible. From the literature of parallel evolutionary algorithms, it is known that the topology is an important factor in the performance of the algorithm because it determines how fast a good solution disseminates to other islands (hives). A small number of islands in a ring topology accelerate the coevolution process, thus requiring a reduced number of cycles/generations.

The Migration gap was set to 10% of Maximum Cycle Number (*MCN*), that is, a migration event occurs at every 600 cycles (recall that the selected *MCN* is 6000 cycles). The migration rate was set to 2 bees, such that a copy of the best bee and a random one are emigrated to the next island of the topology and replace randomly chosen bees of the receiving hive. Two versions were made changing the amount of bees per hive (*Colonysize*). One uses 250 bees per island (HH–ABC_1) and another uses 63 bees per island (HH–ABC_2). The HH–ABC_1 and HH–ABC_2 totalizes 1,000 and 252 bees, respectively. The reason for these two approaches is that the sequential ABC model uses 250 bees and, if we divide this population into four hives, we have 62.5 bees per hive (since only integer numbers make sense, we set to 63 bees per hive). Indices 1 and 2 indicate only different *Colonysize*.

At the lower level, the HH–ABC uses a MS–ABC configuration with one master and 25 slaves. The reason to set this configuration is the same as for the number of islands.

The effect of coevolution between hives is reflected after a migration cycle. When a good quality bee immigrates to a hive, it not only improves the local best solution, but also, through recombination, induces a further improvement of quality in the hive.

The three parallel approaches (MS–ABC, HH–ABC_1 and HH–ABC_2) were run 100 times, keeping fixed all other parameters. Results are not shown here due to space restrictions. However, we observed that the HH–ABC (HH–ABC_1 and HH–ABC_2) approach leads to better results when compared with the MS–ABC (recall that the MS–ABC and sequential versions have exactly the same functionalities). These results strongly suggests that the coevolution effect of the migration between hives is advantageous for the parallel ABC algorithm, and it is used henceforth.

6.2 Benchmark Results and Discussion

In our experiments, four synthetic 27 amino acids-long sequences were used as benchmark, as shown in the columns of the first row of the Tables 1 and 2. These sequences have been used by other researchers for the 3DHP model [13, 5], and to the best of our knowledge, these sequences were used for the first time by [2] for the 3DHP-SC model. Since the ABC algorithm is an stochastic algorithm, experiments were run 100 times with different random seeds. The performance of each approach takes into account the best found solution an the processing time. The MS–ABC model was run with 50 slaves and 250 bees. The HH–ABC was run with 25 slaves per hive (4) and, therefore, each slave computes the fitness function for 10 bees in the HH–ABC_1, and 2 or 3 bees in the HH–ABC_2. Results are shown in Tables 1 and 2.

Comparing the MS–ABC and the ABC (sequential), it is noticed that the quality of solutions was almost the same with statistically insignificant differences. How-

Table 1 Statistical results for benchmark sequences S_1 and S_2

Sequence	S_1 $(PH)^3H^2P^2(HP)^2P^{10}H^2P$		S_2 $PH^2P^{10}H^2P^2H^2P^2HP^2HPH$	
Model	*fitness*	Tp(s)	*fitness*	Tp(s)
ABC (Sequential)	523.27±26.32	58052.15	529.48±23.44	57221.32
MS–ABC	524.15±25.04	1554.38	530.51±22.76	1546.22
HH-ABC$_1$	581.83±35.25	1849.98	673.95±27.42	1848.15
HH-ABC$_2$	517.21±27.04	886.23	588.74±19.86	902.56

Table 2 Statistical results for benchmark sequences S_3 and S_4

Sequence	S_3 $H^4P^5HP^5H^3P^8H$		S_4 $H^3P^2H^4P^3(HP)^2PH^2P^2HP^3H^2$	
Model	*fitness*	Tp(s)	*fitness*	Tp(s)
ABC (Sequential)	621.11±35.74	57358.08	938.36±30.06	57147.24
MS–ABC	620.05±36.86	1547.44	940.91±29.31	1406.21
HH-ABC$_1$	772.36±20.22	1758.01	1002.59±22.21	1788.19
HH-ABC$_2$	711.37±14.48	871.44	989.55±18.59	926.82

ever, the MS–ABC model achieved a significantly speedup ($\sim$40 for sequence S4) and efficiency ($\sim$0.8 for sequence S4). In fact, speedup and efficiency are a direct consequence of the load balancing between master and slaves. Ideally, efficiency should be close to the unity (but not above it). However, in practice, this is not always possible, since processors are not used 100% of time for processing, but also for communication, memory allocation and other tasks of the underlying operating system. This is due to the communication overload caused in the master processor.

The difference between HH–ABC_1 and HH–ABC_2 is the size of their hives. Results indicate that the higher the hive, the better the quality of results. However the computational cost increases linearly with the size of the hive.

7 Conclusions

This paper reports the first work using parallel approaches of the ABC algorithm for the PSP problem using the 3DHP-SC model. PSP is still an open problem in Bioinformatics. Therefore, an important contribution of this work are the results regarding this issue. The use of parallel computing is justified in this work due to the complexity of the fitness function.

Experiments showed that the migration between hives leads to better results due to the effect of coevolution. The *Colonysize* (number of bees) per hive has a significant influence in the quality of solutions. Results suggested that larger colony leads to better results than the smaller ones. Further work will focus on other topologies (i.e. hypercubes, toroidal mesh and fully connected), as well as a deep study of the migration policy between hives. Since the experiments were performed using a cluster computing environment, the communication overhead takes significant influence upon the computational time. Consequently, an appropriate load balance procedure have to be done. To overcome such drawback, future work will consider the use of reconfigurable computing and General-Purpose Graphics Processing Units (GPGPU) to accelerate processing.

We believe that this work is an useful contribution to this area of research, since that the directions pointed out can be useful for other Swarm Intelligence approaches. Future work will also investigate parallel versions of other Swarm Intelligence, such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Firefly Algorithm (FA), so as to compare with the parallel ABC presented in this study. Overall, results were good enough and promising to support the continuity of the work.

References

1. Anfinsen, C.B.: Principles that govern the folding of protein chains. Science 181 (1973)
2. Benítez, C.M.V., Lopes, H.S.: A parallel genetic algorithm for protein folding prediction using the 3DHP side-chain model. In: Proc. of IEEE Congress on on Evolutionary Computation, pp. 1297–1304 (2009)

3. Chidambaram, C., Lopes, H.S.: A new approach for template matching in digital images using an artificial bee colony algorithm. In: World Congress on Nature and Biologically Inspired Computing, pp. 146–151 (2009)
4. Dill, K.A., Bromberg, S., Yue, K., et al.: Principles of protein folding - a perspective from simple exact models. Protein Science 4(4), 561–602 (1995)
5. Guo, Y.-Z., Feng, E.-M.: The simulation of the three-dimensional lattice hydrophobic-polar protein folding. Journal of Chemical Physics 125, 234–703 (2006)
6. Karaboga, D.: An idea based on honey bee swarm for numerical optimization. Technical report, Department Computer Engineering Department, Erciyes University (2005)
7. Li, M.S., Klimov, D.K., Thirumalai, D.: Folding in lattice models with side chains. Computer Physics Communications 147(1), 625–628 (2002)
8. Lopes, H.S.: Evolutionary algorithms for the protein folding problem: A review and current trends. In: Computational Intelligence in Biomedicine and Bioinformatics, vol. I, pp. 297–315. Springer, Heidelberg (2008)
9. Lopes, H.S., Scapin, M.P.: An enhanced genetic algorithm for protein structure prediction using the 2D hydrophobic-polar model. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 238–246. Springer, Heidelberg (2006)
10. Nakrani, S., Tovey, C.: On honey bees and dynamic server allocation in internet hosting centers. Adaptive Behavior 12(3-4), 223–240 (2004)
11. Pawar, P.J., Rao, R.V., Shankar, R.: Multi-objective optimization of electro-chemical machining process parameters using artificial bee colony (ABC) algorithm. In: Advances in Mechanical Engineering (AME 2008) (December 2008)
12. Sato, T., Hagiwara, M.: Bee system: Finding solution by a concentrated search. In: Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, vol. 4(C), pp. 3954–3959 (1997)
13. Unger, R., Moult, J.: A genetic algorithm for 3D protein folding simulations. In: Proc. 5^{th} Annual Int. Conf. on Genetic Algorithms, pp. 581–588 (1993)

Parallel Constraint-Based Local Search on the Cell/BE Multicore Architecture

Daniel Diaz, Salvador Abreu, and Philippe Codognet

Abstract. We investigate the use of the Cell Broadband Engine (Cell/BE) for Combinatorial Optimization applications. We present a parallel version of a constraint-based Local Search algorithm which was chosen because it fits very well the Cell/BE architecture since it requires neither shared memory nor communication between processors. The performance study on several large optimization benchmarks shows mostly linear time speedups, even sometimes super-linear. These experiments were done on a Dual-Cell IBM Blade with 16 processors. Besides getting speedups, the execution times exhibit a much smaller variance, which benefits applications where a timely reply is critical.

1 Introduction

The Cell Broadband Engine (Cell/BE) has proved its suitability for graphical applications and scientific calculations [17], due to an innovative multicore architecture with its 8 independent, specialized processing units. However, its ability to perform well for general-purpose applications is questionable, as it is very different from classical homogeneous multicore processors from Intel, AMD or Sun (Niagara), or even IBM's Power6 and 7. In this paper we investigate the use of Cell/BE for Combinatorial Optimization: this is a first step towards a large-scale implementation on a massively parallel architecture, with reduced communication.

Daniel Diaz
University of Paris 1-Sorbonne, France
e-mail: `Daniel.Diaz@univ-paris1.fr`

Salvador Abreu
Universidade de Évora and CENTRIA FCT/UNL, Portugal
e-mail: `spa@di.uevora.pt`

Philippe Codognet
JFLI, CNRS / UPMC / University of Tokyo, Japan
e-mail: `Philippe.Codognet@lip6.fr`

We have developed a parallel extension of a constraint-based Local Search algorithm called Adaptive Search (AS) [4]. This metaheuristic is quite efficient in practice, compared to classical propagation-based constraint solvers, especially for large problems. We implemented a parallel version of Adaptive Search for Cell (AS/Cell). To assess the viability of this approach, we experimented AS/Cell on several classical benchmarks from CSPLib [8]. These structured problems are abstractions of real problems and therefore representative of real-life applications; they are classically used for benchmarking new methods. The results for the parallel Adaptive Search method show a good behavior when scaling up the number of cores: speedups are practically linear, especially for large-scale problems and sometimes we reach super-linear speedups because the simultaneous exploration of different subparts of the search space may converge faster towards a solution.

Another interesting aspect is that all experiments show a better robustness of the results when compared to the sequential algorithm. Because Local Search (LS) makes use of randomness for the diversification of the search, execution times vary from one run to another. When benchmarking such methods, execution times have to be averaged over many runs (we take the average of 50 runs). Our implementation results show that with AS/Cell running on 16 cores, the difference between the minimum and maximum execution times, as well as the overall variance of the results, decreases significantly w.r.t. the reference sequential implementation. Execution times become more predictable: this is a clear advantage for real-time applications with bounded response time requirements.

The rest of this article is organized as follows: Section 2 presents Parallel Local Search. Section 3 discusses the Adaptive Search algorithm and its parallel version is presented in Section 4. A performance analysis is shown in Section 5. The robustness of the method is studied in Section 6. A short conclusion ends this paper.

2 Parallel Local Search

Parallel implementation of local search metaheuristics have been studied since the early 90's, when multiprocessor machines started to become widely available, see [19] for a general survey and concepts, or [15] for basic parallel version of Tabu search, simulated annealing, GRASP and genetic algorithms. With the availability of clusters in the early 2000's, this domain became active again [5, 2]. Apart from domain-decomposition methods and population-based method (such as genetic algorithms), one usually distinguish between single-walk and multiple-walk methods for LS. Single-walk methods consist in using parallelism inside a single search process, e.g. for parallelizing the exploration of the neighborhood, see for instance [10] for such a method making use of GPUs for the parallel phase. Multiple-walk methods (also called multi-start methods) consist in developing concurrent explorations of the search space, either independently or cooperatively with some communication between concurrent processes. A key point is that independent multiple-walk methods are the most easy to implement on parallel computers and can lead to linear speed-up if the solutions are uniformly distributed in the search space and if the method is able to diversify correctly [1]. Sophisticated cooperative strategies for multiple-walk methods can be devised by using solution pools [6], but requires shared-memory or emulation of central memory in distributed clusters, impacting thus on performances.

In Artificial Intelligence, parallel implementation of search algorithms has a long history [7]. For Constraint Satisfaction Problems (CSP), early work has been done in the context of Distributed Artificial Intelligence and multi-agent systems [20], but these methods cannot lead to efficient algorithms and cannot compete with good sequential implementations. Only very few implementations of efficient constraint solvers on parallel machines have been reported, most notably [16] for shared-memory architectures and recently [12] who proposes a distributed extension of the Comet LS solver for clusters of PCs. In the domain of SAT solving (boolean satisfiability), most parallel implementations have targeted multi-core architectures with shared memory [9, 3, 18]. Very recently, [14] extended a solver for PC cluster architectures by using a hierarchical shared memory model in order to to minimize communication between independent machines.

3 The Adaptive Search Algorithm

Over the last decade, the application of LS techniques in the CSP community has started to draw some interest. A generic, domain-independent LS method named Adaptive Search (AS) was proposed in [4]. AS is a meta-heuristic that takes advantage of the structure of the problem to guide the search more precisely than a unique global cost function like the number of violated constraints. This method is generic, can be applied to a large class of constraints (e.g. linear and non-linear arithmetic constraints, symbolic constraints...) and intrinsically copes with over-constrained problems. The input is a problem in CSP format, that is, a set of variables with their domains of possible values and a set of constraints over these variables. For each constraint, an "error function" needs to be defined; it will give, for each tuple of variable values, a quantification of how much the constraint is violated. For instance, the error function associated with an arithmetic constraint $X < c$, for a given constant $c \geq 0$, could be $max(0, X - c)$. Adaptive Search relies on *iterative repair*, based on variable and constraint errors, seeking to reduce the error on the worst variable. The basic idea is to compute the error function for each constraint and then combine, for each variable, the errors of all constraints in which it appears, thereby projecting constraint errors onto the relevant variables. Finally, the variable with the highest error will be designated as the "culprit" and its value will be modified. In this second step, the *min-conflict* heuristic [13] is used to select the value in the variable domain which is the most promising, that is, the value for which the total error in the next configuration is minimal. To prevent being trapped in local minima, the Adaptive Search method includes a short-term memory mechanism in the spirit of Tabu Search (variables can be marked Tabu and "frozen" for a number of iterations). It also integrates reset transitions to escape stagnation around local minima. A reset consists in assigning fresh random values to some variables (randomly chosen) and is guided by the number of variables being marked Tabu. It is also possible to restart from scratch when the number of iterations becomes too large (this sort of "reset over all variables" is guided by the number of iterations).

For instance, consider an n-ary constraint $c(X_1, \cdots X_n)$ and associated variable domains $D_1, \cdots D_n$. An error function f_c associated to the constraint c is a function from $D_1 \times \cdots \times D_n$ such that $f_c(X_1, \cdots X_n)$ has value zero iff $c(X_1, \cdots X_n)$ is satisfied. The error function is used as a heuristic to represent the degree of satisfaction of a constraint and thus gives an indication on how much the constraint is violated. This is very similar to the notion of "penalty functions" used in continuous global optimization. This error function can be seen as (an approximation of) the distance of the current configuration to the closest satisfiable region of the constraint domain. Since the error is only used to heuristically guide the search, we can use any approximation when the exact distance is difficult (or even impossible) to compute. Adaptive Search is a simple algorithm (see Algorithm 4) but it turns out to be quite effective in practice [4]. Considering the complexity/efficiency ratio, this can be a very effective way to implement constraint solving techniques, especially for "anytime" algorithms where (approximate) solutions have to be computed within a bounded amount of time.

4 A Parallel Version of Adaptive Search on Cell/BE

We now present AS/Cell: our implementation of the Adaptive Search algorithm on the Cell/BE. We do not detail the Cell/BE processor here, but some features deserve mention since they strongly shape what applications stand a chance to succeed when ported:

- A hybrid multicore architecture, with a general-purpose "controller" processor (the PPE, a PowerPC instance) and eight specialized streaming SIMD processors ("Synergistics Processing Elements" or SPEs).
- Two Cell/BE processor chips may be linked on a single system board to appear as a multiprocessor with 16 SPEs.
- The SPEs are connected via a very high-bandwidth internal bus, the EIB.
- The SPEs may only perform computations on their local store (256KB for code+data).
- The SPEs may access main memory and each other's local store via DMA operations.

The interested reader can refer to the IBM Redbook [17] for further information, as well as the performance and capacity tradeoffs which affect Cell/BE programs. A-priori, the Adaptive Search algorithm seems to fit the requirements fairly well, justifying our choice.

Algorithm 1 Adaptive Search Base Algorithm

Input: problem given in CSP format:	some tuning parameters:
• set of variables X_i with their domains	• TT: number of iterations a variable is frozen
• set of constraints C_j with error functions	• RL: number of frozen variables triggering a reset
• function to project constraint errors on vars	• RP: percentage of variables to reset
• (positive) cost function to minimize	• MI: maximal number of iterations before restart
	• MR: maximal number of restarts

Output: a solution if the CSP is satisfied or a quasi-solution of minimal cost otherwise.

```
1: $Restart \leftarrow 0$
2: repeat
3:     $Restart \leftarrow Restart + 1$
4:     $Iteration \leftarrow Tabu\_Nb \leftarrow 0$
5:     Compute a random assignment $A$ of variables in $V$
6:     $Opt\_Sol \leftarrow A$
7:     $Opt\_Cost \leftarrow cost(A)$
8:     repeat
9:         $Iteration \leftarrow Iteration + 1$
10:        Compute errors of all constraints in $C$ and combine errors on each variable
11:                          $\triangleright$ (by considering only the constraints in which a variable appears)
12:        select the variable $X$ (not marked Tabu) with highest error
13:        evaluate costs of possible moves from $X$
14:        if no improvement move exists then
15:            mark $X$ as Tabu until iteration number: $Iteration + TT$
16:            $Tabu\_Nb \leftarrow Tabu\_Nb + 1$
17:            if $Tabu\_Nb \geq RL$ then
18:                randomly reset $RP$ % variables in $V$ (and unmark those Tabu)
19:            end if
20:        else
21:            select the best move and change $X$, yielding the next configuration $A'$
22:            if $cost(A') < Opt\_Cost$ then
23:                $Opt\_Sol \leftarrow A \leftarrow A'$
24:                $Opt\_Cost \leftarrow cost(A')$
25:            end if
26:        end if
27:    until a solution is found or $Iteration \geq MI$
28: until a solution is found or $Restart \geq MR$
29: $output(Opt\_Sol, Opt\_Cost)$
```

The basic idea in extending AS for parallel implementation is to have several distinct parallel search engines for exploring simultaneously different parts of the search space, giving thus an independent multiple-search version of the sequential AS. This is very natural to achieve with the AS algorithm: one just needs to start each engine with a different, randomly computed, initial configuration, that is, a different assignment of values to variables. Subsequently, each "AS engine" can perform the sequential algorithm independently and, as soon as one processor finds a solution or when all have reached the maximum number of allowed iterations, all processors halt and the algorithm finishes. The Cell/BE processor architecture is mapped onto the task structure, with a controller thread on the PPE and a worker thread on each SPE.

1. The PPE gets the real time $T0$, launches a given number of threads (SPEs), each with an identical SPE context, and then waits for a solution.
2. Each SPE starts with a random configuration (held entirely in its local storage) and improves it step by step, applying the algorithm of Section 3.
3. As soon as an SPE finds a solution, it sends it to main memory (using a DMA operation) and informs the PPE.
4. The PPE then instructs all other SPEs to stop, and waits until all SPEs have done so (join). After that, it gets the real time $T1$ and provides both the solution and the execution time $T = T1 - T0$ (this is the real elapsed time since the beginning of the program until the join including SPEs initialization and termination).

This simple parallel version seems feasible and does not require complex data structures (e.g. shared tables). Note that SPEs do not communicate, only doing so with the PPE upon termination: each SPE works blindly on its configuration until it reaches a solution or fails. We plan to further extend this algorithm with communication between SPEs of some information about partial solutions, but always with the aim of limiting data communication as much as possible. It remained to be seen whether the space limitations of the Cell/BE processor, in particular the size of the SPE local stores, are not crippling in terms of problem applicability. We managed to fit both the program and the data in the 256KB of local store of each SPE, even for large benchmarks. This turned out possible for two reasons: the relative simplicity of the AS algorithm, and the compactness of the encoding of combinatorial search problems as a CSP, that is, variables with finite domains and many predefined constraints, including arithmetic ones. This is especially relevant when compared, for instance, to a SAT encoding where only boolean variables can be used and each constraint has to be explicitly decomposed into a set of boolean formulas yielding problem formulations which easily reach several hundred thousand literals.

In short, the Adaptive Search method requirements make a good match for the Cell/BE architecture: *little data but a lot of computation*.

5 Performance Evaluation

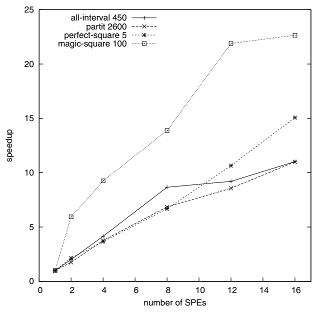
We now present and discuss our assessment of the performance of the initial implementation of AS/Cell. The code running on each SPE is a port of that used in [4], an implementation of Adaptive Search specialized for permutation problems. It should be noted that no code optimizations have been made to benefit from the peculiarities of Cell/BE (namely SIMD vectorization, loop unfolding and parallelization, branch removal...). Our initial measurements, made on the IBM system simulator, lead us to believe it is reasonable to expect a significant speedup when these aspects are taken into account, as the SPEs are frequently stalled when executing the present code. In order to assess the performance of AS/Cell, we use a pertinent set of classical benchmarks from CSPLib [8] consisting of:

- `all-interval`: the All-Interval Series problem (`prob007` in CSPLib).
- `perfect-square`: the Perfect-Square placement problem (`prob009` in CSPLib).

- `partit`: the Number Partitioning problem (`prob049` in CSPLib).
- `magic-square`: the Magic Square problem (`prob019` in CSPLib).

Although these are academic benchmarks, they are abstractions of real-world problems and representative of actual combinatorial optimization applications. These benchmarks involve a very large combinatorial search space, e.g. the 100×100 magic square requires 10000 variables whose domains range over 10000 values.

The experiment has been executed on an IBM QS21 dual-Cell blade system (times are measured in seconds). Since Adaptive Search uses random configurations and progression, each benchmark was executed 50 times. The resulting execution time is the average of the 50 runs after removing both the lowest and the highest times.



We ran a series of tests, varying the number of SPEs from 1 to 16. The figure depicts the evolution of the speedup w.r.t the number of SPEs for the hardest instance of each problem.

Concerning raw performance, it is worth noticing that AS/Cell performs very well even on difficult problems. As case study, consider the magic squares problem (one of the most challenging: most solvers cannot even handle instances larger than 20×20). With 1 SPE, the average time to solve magic square 100×100 is around 3 minutes. Using 16 SPEs, AS/Cell drastically reduces the computation time, bringing it to barely over 8 seconds. Moreover, these results are obtained with a straightforward port of the AS sequential code, not yet specialized to benefit from the potential of Cell/BE.

We performed some initial code profiling and datapath-level simulations to check the low-level performance and effective hardware utilization. At the SPE level, our initial port yields a CPI of `2.05`, which puts us at about one third of the performance which we can expect to attain. Approximately half the cycles in SPE executions are spent in stalls, of which about 60 % are branch misses while the remaining 40 % are spent on dependency stalls. Again, these results are not surprising, since we did a direct port of the existing, sequential code. This leads us to expect a significant performance increase from execution analysis and code reorganization. We also plan on experimenting with SIMD vectorization, loop unrolling, branch removal and other techniques to further improve performance.

Our first assessment clearly shows that the Adaptive Search method is a good match for the Cell/BE architecture: this processor is definitely effective in solving nontrivial highly combinatorial problems.

6 Robustness Evaluation

In the previous section each benchmark was run 50 times and the average time was taken, leaving the extreme values out. This is a classical approach and gives us a precise idea of the general behavior of AS/Cell. In this section, we study the degree of variation between the various runs, and how AS/Cell can influence it. One way to measure dispersion is to consider, for each instance, the longest execution time among the 50 runs. This is interesting for situations such as real-time applications since it represents the "worst case" one can encounter: too high a value can even inhibit use in time-critical situations.

This table summarizes the results, focusing on the worst case.[1] This evaluation uncovers a significant improvement: the obtained speedup is always better than the one obtained in the average case. For instance, with 16 SPEs, the worst case is improved by factor 26 for `all-interval 450` and by a factor 500 for `magic-square 100` (the corresponding average time speedups are 11 and 22.6). Clearly, AS/Cell greatly narrows the range of possible execution times for a given problem. Another way to measure this is to consider the standard deviation: the table below charts the evolution of the standard deviation of the execution times for the hardest instance of each problem varying the number of SPEs.

Problem Instance	Speedup for k SPEs				
	2	4	8	12	16
`all-interval 450`	2.4	4.7	15	11	**26**
`partit 2600`	1.2	3.6	7.2	12	**17**
`perfect-square 5`	2.1	3.9	7.1	9.7	**17**
`magic-square 100`	63	150	160	540	**510**

The standard deviation rapidly decreases as more SPEs are used. For instance, considering magic square 100×100, the standard deviation decreases from 916 to less than 4. This amounts to a dramatic performance improvement for the worst case scenario. Take magic square 100×100: with 1 SPE execution needs 2.5 hours at worst. When using 16 SPEs, this drops to 18 seconds.

Problem Instance	Standard deviation with k SPEs					
	1	2	4	8	12	16
`all-interval 450`	891	460	224	65	61	40
`partit 2600`	24	16	7	3	2	2
`perfect-square 5`	122	54	27	16	10	6
`magic-square 100`	916	19	13	10	3	4

AS/Cell limits the dispersion of the execution times: we say that the multicore version is *more robust* than the sequential one. The execution time is more predictable from one run to another in the multicore version, and more cores means more robustness. This is a crucial usability feature for real-time systems or even for some interactive applications such as games. To further test this idea we tried a slight variation of the method which consists in starting all SPEs with the *same* initial configuration (instead of a random one). Each SPE then diverges according to its own internal random choices. The results of this experiment show that the overall behavior is practically the same as the original method, only a little slower: on the average less than 10%. This slowdown was to be expected because the search has less diversity to start with, and therefore might take longer to explore a search space that contains a solution. However, this limited slowdown shows that the method is intrinsically robust, can restore diversification and again take advantage of the parallel search in an efficient manner.

[1] Due to space limitation we only show the values for the hardest instance of each problem.

7 Conclusion

We presented a simple yet effective parallel adaptation of the Adaptive Search algorithm to the Cell/BE architecture to solve combinatorial problems. We chose to target the Cell/BE because of its promise of high performance and, in particular, to experiment with its heterogeneous architecture, which departs significantly from most multicore architectures with shared memory. The implementation drove us to minimize communication of data between processors and between a processor and the main memory. We view this experiment as the first step towards a large-scale implementation on a massively parallel architecture, where communication costs are at a premium and should be eschewed.

The experimental evaluation we carried out on a dual-CPU blade with 16 SPE cores indicates that linear speedups can be expected in most cases, and even some situations of super-linear speedups are possible. Scaling the problem size seems never to degrade the speedups, even when dealing with very difficult problems. We even ran a reputedly very hard benchmark with increasing speedups when the problem size grows. An important, if somewhat unexpected, fringe benefit is that the worst case execution time gets even higher speedups than the average case. This characteristic opens up several domains of application to the use of combinatorial search problem formulations: this is particularly true of real-time applications and other time-sensitive uses, for instance interactive games.

Recall that Adaptive Search is an "anytime" method: it is always possible to interrupt the execution and obtain the best pseudo-solution computed so far. Regarding the issue, this method can easily benefit from an architecture such as Cell/BE: when running several SPEs in parallel, the PPE simply queries each SPE to obtain its best pseudo-solution (together with the corresponding cost) and then chooses the best of these best. Another good property of the Cell/BE organization is that the only data a SPE needs to pass, when copying, is the current configuration (a small array of integers) and its associated cost, which can be done very efficiently.

Our early results clearly demonstrate that heterogeneous architectures with internal parallelism, such as Cell/BE, have a significant potential to make good on solving combinatorial problems. Concerning further development, we plan to work along two directions: to optimize the Cell/BE code following the recommendations of [17] and to experiment with several forms of communication among the processors involved in a computation. The extension of the algorithm beyond a single dual-Cell blade is also in our plans, in particular to extend our approach with cluster communication mechanisms, such as MPI or the RDMA-based Fraunhofer Virtual Machine [11].

Acknowledgements. The equipment used in this work was provided as a grant from the IBM Corporation Shared University Research (SUR) program awarded to CENTRIA and U. of Évora. Acknowledgements are due to the FCT/Egide Pessoa grant "CONTEMP – CONTraintes Executes sur MultiProcesseurs."

References

1. Aiex, R.M., Resende, M.G.C., Ribeiro, C.C.: Probability distribution of solution time in grasp: An experimental investigation. Journal of Heuristics 8(3), 343–373 (2002)
2. Alba, E.: Special issue on new advances on parallel meta-heuristics for complex problems. Journal of Heuristics 10(3), 239–380 (2004)
3. Chu, G., Stuckey, P.: A parallelization of minisat 2.0. In: SAT race (2008)
4. Codognet, P., Diaz, D.: An efficient library for solving csp with local search. In: Ibaraki, T. (ed.) MIC 2003, 5th International Conference on Metaheuristics (2003)
5. Crainic, T., Toulouse, M.: Special issue on parallel meta-heuristics. Journal of Heuristics 8(3), 247–388 (2002)
6. Crainic, T.G., Gendreau, M., Hansen, P., Mladenovic, N.: Cooperative parallel variable neighborhood search for the median. Journal of Heuristics 10(3), 293–314 (2004)
7. de Kergommeaux, J.C., Codognet, P.: Parallel logic programming systems. ACM Computing Surveys 26(3), 295–336 (1994)
8. Gent, I.P., Walsh, T.: Csplib: A benchmark library for constraints. In: Jaffar, J. (ed.) CP 1999. LNCS, vol. 1713, pp. 480–481. Springer, Heidelberg (1999)
9. Hamadi, Y., Jabbour, S., Sais, L.: Manysat: a parallel sat solver. Journal on Satisfiability, Boolean Modeling and Computation 6, 245–262 (2009)
10. Van Luong, T., Melad, N., Talbi, E.-G.: Parallel local search on gpu. Technical Report RR 6915, INRIA, Lille, France (2009)
11. Machado, R., Lojewski, C.: The Fraunhofer virtual machine: a communication library and runtime system based on the RDMA model. Computer Science-Research and Development 23(3), 125–132 (2009)
12. Michel, L., See, A., Van Hentenryck, P.: Distributed constraint-based local search. In: Benhamou, F. (ed.) CP 2006. LNCS, vol. 4204, pp. 344–358. Springer, Heidelberg (2006)
13. Minton, S., Johnston, M.D., Philips, A.B., Laird, P.: Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. Artif. Intell. 58(1-3), 161–205 (1992)
14. Ohmura, K., Ueda, K.: c-sat: A parallel sat solver for clusters. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 524–537. Springer, Heidelberg (2009)
15. Pardalos, P.M., Pitsoulis, L.S., Mavridou, T.D., Resende, M.G.C.: Parallel search for combinatorial optimization: Genetic algorithms, simulated annealing, tabu search and grasp. In: Ferreira, A., Rolim, J.D.P. (eds.) IRREGULAR 1995. LNCS, vol. 980, pp. 317–331. Springer, Heidelberg (1995)
16. Perron, L.: Search procedures and parallelism in constraint programming. In: Jaffar, J. (ed.) CP 1999. LNCS, vol. 1713, pp. 346–360. Springer, Heidelberg (1999)
17. IBM Redbooks. Programming the Cell Broadband Engine Architecture: Examples and Best Practices. Vervante (2008)
18. Schubert, T., Lewis, M.D.T., Becker, B.: Pamiraxt: Parallel sat solving with threads and message passing. Journal on Satisfiability, Boolean Modeling and Computation 6, 203–222 (2009)
19. Verhoeven, M., Aarts, E.: Parallel local search. Journal of Heuristics 1(1), 43–65 (1995)
20. Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: The distributed constraint satisfaction problem: Formalization and algorithms. IEEE Transactions on Knowledge and Data Engineering 10(5), 673–685 (1998)

Analysis of Huge Volume of Documents on a Hybrid HPC Cluster

Stefan-Gheorghe Pentiuc and Ioan Ungurean

Abstract. The analysis of huge volumes of data has always required high computational ability. Implementing the clustering and classification algorithms upon an ordinary computer will not be efficient, since this shows limitations as regards the computational ability and available memory. Within this paper, an algorithm of clustering the huge volumes of data on a hybrid HPC cluster, is proposed. This cluster has a RoadRunner architecture. It includes 48 QS22 blade servers, each of them having 2 PowerXCell 8i processors. The algorithm proposed within the present paper will carry out the clustering of a number of documents, by using a parallel version of the k-means algorithm, implemented upon the RoadRunner hybrid architecture.

1 Introduction

Currently, the Internet containes billion pages and their number continues to grow. The analysis and review of the information contained into these documents comes to be very difficult without the help of the methods and algorithms from the unsupervised context of pattern recognition. One example is the algorithm able to determine a partition of the set of documents based on a particular similarity measure. Since the data sets are of gigabytes order, the sequential algorithms cannot efficiently accomplish the task, because one processor has usually a limited memory. The parallel algorithms [1] can solve this issue, by using a high number of processors, each of them with its own memory.

The notion of pattern in pattern recognition is defined as a mathematical representation of an entity by the values of its attributes[2]. In the document clustering, the patterns represents a document clustering meets the following stages [3]:

- representing the documents by patterns;
- defining a similarity function;
- determine a partition of the documents set (clustering).

Representing the documents as patterns means to define the features with discriminative potential, and to determine the numerical values of these features in the

Stefan-Gheorghe Pentiuc · Ioan Ungurean
Stefan cel Mare University of Suceava, str.Universitatii nr.13, RO-720229 Suceava Romania
e-mail: pentiuc@eed.usv.ro, ioanu@eed.usv.ro

case of each document. Finally all the documents will be represented by vectors belonging to the same vector space.

The k-means algorithm [4] is one of the most significant data clustering algorithms, which can be used for document clustering, but depending upon the way of representing these documents.

As concerns the document clustering, a parallel k-means algorithm is proposed within this paper, which runs over a couple of blade QS22 servers. These blade servers include two PowerXCell 8i processors, the new generation of processors based upon the Cell Broadband Engine (Cell/B.E.) Architecture [5].

2 The Representation of Documents

The first step on performing the process of document clustering consists of representing the documents belonging to the input data set, under a numerical form, so as to apply the algorithms of data clustering.

A method of representing the documents is based on a dictionary of terms or an area of terms. Let be p the number of the terms in the dictionary. For each document will be computed the frequency of occurrence in the document of each term from the dictionary. The resulted p values are the p components of a pattern vector that represents the document

$$X = \left\{x_1, x_2, x_3, \ldots x_p\right\} \tag{1}$$

In this way each document may be viewed also by a point in a p-dimensional pattern space.

Let n the number of documents taken into consideration and represented by patterns. The E set of patterns is represented by a n p matrix, Each row of the matrix will correspond to a document, and each column will signify a term of the dictionary. The next step on carrying out the process of document clustering is defining a similarity measure amongst two documents, by the help of numerical representation of documents.

3 Similarity Measure

The most used method of measuring the similarity within data clustering is represented by a function decreasing with distance between the pattern points.

The similarity measure is a function

$$\varphi : E \times E \to R^+ \tag{2}$$

that is required to satisfy

$$\varphi(x,y) = \varphi(y,x), \forall (x,y) \in E \times E \tag{3}$$

$$\varphi(x,x) = \varphi(y,y) = max\{\varphi(x,y), \forall (x,y) \in E \times E\} \tag{4}$$

As is shown in [7], usually in the p-coordinates space representing the documents the similarity between two points was considered to be the function

$$\varphi(x,y) = cos(x,y), \forall(x,y) \in E \times E. \tag{5}$$

This similarity function, known as cosine similarity, has the advantage that that similarity amongst the two vectors does not depend upon their size: the less the angle between these two vectors is, the more the cosines of this angle will reach 1.

The similarity among two pattern vectors x and y representing the documents having p components as in (1) may be computed as follows

$$\varphi(x,y) = \frac{\sum_{i=1}^{p} x_i y_i}{\sqrt{\sum_{i=1}^{p} x_i^2}\sqrt{\sum_{i=1}^{p} y_i^2}} \tag{6}$$

where the features xi and yi are calculated like in [7] as tfi /tfmax,with tfi the frequency of term from dictionary in the document, tfmax is the maximum frequency of the term in all the documents. This method might be used in order to classify the documents, by measuring the similarity of them towards a standard vector.

4 The k-Means Algorithm

Concerning the situation presented, using the k-means algorithm for a large data set was approached. This algorithm has to run in parallel on a couple of QS22 nodes [5]. In this case, these will be distributed towards k nodes. The k-means has proven to be an algorithm useful to clustering a data set into k clusters.

Let the pattern set

$$E = \{x_1, x_2, x_3, ...x_p\} \tag{7}$$

contains n patterns of dimension p representing n documents, then the clustering process has to group n patterns into k clusters. The essential of the k-means algorithm is to group the patterns around the k centers cj of clusters, so that the similarity amongst each pattern and the center of its cluster will be maximized:

The algorithm is convergent because the assignment of the pattern to the near-est center leads to a minimum of intra-cluster inertia, and maximizes the criterion (3).

The main characteristic of k-means algorithm is the inherent parallelism. For large sets of patterns a massive computing effort calculation is needed to deter-mine, at each iteration, the distance between the n patterns and the k centers.

Idea is to share the computing tasks on multiple processors taking into account that the data communication between processors (Tcommunication ¿¿ Tcomputing). For greater efficiency is needed to minimize communication. The implementation may be on a computer network, [9],[10] or on a high coupled system, an IBM cluster from the our university High Performance Computing Center.

A parallelization of this algorithm means to divide the computing effort into *m*+1 processes in which one is called *root* and the other *m* processes will be called *node*.

The description of the algorithm is presented in a chronological order of an iteration such as follows:

```
Root process - initial stage
*) divide the training set into $m$ subsets $Fl$, with $l$=1,$m$ and send to each Node process a subset containing about $n/m$ patterns
*) initializes the list of $k$ centers $cj$ and then broadcasts it to other processes
*) waits for data from all the $m$ Node processes
Node process "l"
*) receives the subset $Fl$ and stores it locally
*) receives the list of centers $c$q, $q$=1,$k$
*) initializes the number of patterns in the future local clusters $nlj$=0, $q$=1,$k$
*) initializes the partial gravity centers of clusters $g$lq=0, $q$=1,$k$
donel=true
for i = 1, card($Fl$) execute *) compute k distances $d(x$i,$c$q), $q$=1,$k$
    *) assign document $xi$ to the cluster $j$, if $d(x$i,$c$j) = min {$d(x$i,$c$q)}, $q$=1,$k$
    $g$lj = $g$lj + $x$i
    $nlj$= $nlj$ + 1
    if document $xi$ was not previously in cluster $j$ then donel=false end
end
Send to the root $g$lj, $nlj$ and donel
```

After sending the data the l Node process waits for other data from the Root. When all the Node processes sent data to the Root this process will continue to execute the following:

```
Root process - recomputing centers stage
if done1 and done2 and .... and donem = true then stop end
$c$q=0 , $q$=1,$k$
for q = 1, k execute
    nq=0
    for $l$=1,m execute
        $c$q= $c$q + $g$lq
        nq = nq + nlq
    end
    $c$q= $c$q / nq
end
Send new centers $c$q to initiate a new iteration on the $m$ Node processes.
*) waits for data from all the $m$ Node processes
```

It may observe that in each iteration there is two data communication between Root and node processes. Each process node receives a list of k centers, computes the new groups of the patterns stored locally and send information to the root node (a boolean value of variable donel, k integers nlj and k vectors glj containing the sum of p – coordinates values of the nlj patterns stored locally and belonging to the cluster Cj). After receiving these values from all the m processes the Root node check if all Boolean values *donel* are true. If yes the clustering is finished and Root stops the calculation. If not, the Root process recomputes the centers based on the partial information received from Node processes and resend the list of k new centers to all the m node processes. This version of the algorithm exploits the fact that the $n \times k$ distances are computed parallel on m processors reducing the calculus time by m times.

5 Developing Applications for the RoadRunner HPC Cluster

For running in parallel this algorithm, upon a couple of QSS22 nodes, the architecture of these computing nodes has to be taken into consideration. The PowerXCell 8i processor on the QS22 has nine heterogeneous cores connected by a 288GB/s on-chip bus, and has been designed to be highly efficient, if considering the power point of view. The PowerXCell 8i processor is an asymmetric multi-core processor, which is optimized for parallel processing and streaming applications. The PowerXCell 8i processor includes a Power Processor Element (PPE) and eight highly optimized enhanced double precision (eDP) SIMD engines, called Synergistic Processor Elements (SPE). The PPE is intended to run the operating system and coordinate computation. Each SPE is able to perform mostly the same as, or better than, a General Purpose Processor (GPP) with SIMD running at the same frequency [5].

Taking into consideration the programmer's point of view, a QS22 node presents two PowerXCell 8i processors. It only has access to a PPE, from which it might create 16 threads running in parallel on those 8 SPE, belonging to the two PowerXCell 8i processors. In this way, the application that runs on a couple of nodes QS22 will prove two levels of parallelization: the first level is accomplished by means of MPI libraries, so as to run in parallel on QS22 nodes; the second level is carried out by means of DaCS libraries, in order to run in parallel on each QS22 node, by using those sixteen SPE kernels. The PPE processors are only used for the distribution and taking over the data from/to those sixteen SPE processors of each node.

The k-means parallel proposed algorithm was programmed to work on m Node processes implemented on m nodes QS22.

One might notice that documents are equally divided to those m nodes, each subset $\boldsymbol{Fl}$ is stored on a QS22 node. Here the data are again divided to those SPE processors. The PPE processor is only used for the communication with the other QS22 nodes and for transmitting to the SPE processors the information needed to run in parallel the algorithm.

The application is performed so that at its execution, a thread is started on each SPE core. This thread can run operations just on data being stored in the local memory of 256KB. Due to this aspect, if accomplishing some operations over shared main memory is wanted, the data will be transferred by DMA in the local memory associated to SPE core; the specific processing is carried out and the result will be transferred in the main memory, so as to be processed by the PPE core. Synchronization between PPE core and SPE cores is very important. In this way, at starting the execution line on SPE core, it waits by means of a mailbox for commands from PPE core.

6 Experimental Results

In order to test the proposed algorithm, the input data sets offered by UCI Machine Learning Repository [8] were used. It was used the NIPS full papers, which include 1500 documents with 12419 terms of dictionary size, the KOS blog entries that contain 3430 documents with 6906 of dictionary size.

The tests were carried out on a cluster foreseen with 48 QS22 nodes. Regarding these tests, the effective time of computing was measured, but the time necessary for reading the input dataset of the file and the distribution of this dataset towards the computing nodes were not taken into consideration. During tests, two MPI processes run on each node (one for each PowerXCell 8i processor of the QS22 blade server). For instance, if two nodes are used for a test, then the application will execute four MPI processes. The input dataset is shared equally to each MPI process. Since every MPI process corresponds to a CELL processor, the distribution within a MPI process of the input data set and of the calculus will be performed at those eight SPE processors specific to PowerXCell 8i processor, attached to the current MPI process.

For each input set, grouping of documents was accomplished into 10, 25, 50 or 100 clusters. For any of these groups, the application was run by using a MPI process (only one processor of a QS22 node was used), two MPI processes (a QS22 node) and 4,8,16,32,64 or 84 MPI processes. The results of the tests are shown for 26 iterations [4].

The experimental results for the KOS input set are illustrated in Fig. 1.

Considering these results, one might analyze that by increasing the number of MPI processes, a significant reducing specific to execution time takes place. Subsequently, the execution time will increase if the number of clusters, where documents are grouped, is increased. Fig. 1 shows also the reducing of execution time

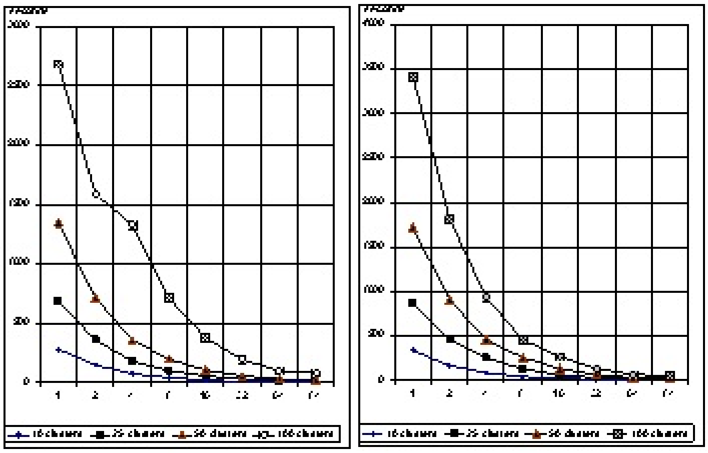


Fig. 1 Experimental results of the KOS dataset (first) and NIPS dataset (second)

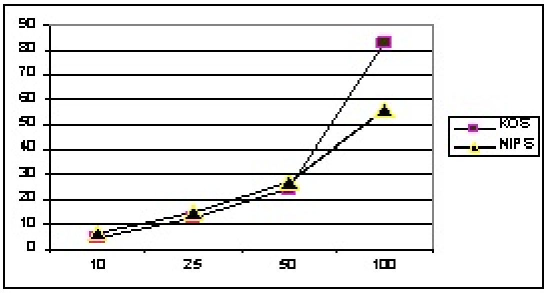


Fig. 2 Experimental results on a 84 processors configuration for a variable number of document clustering with the both datasets

for documents' grouping within the KOS input dataset into 25 clusters, by means of increasing the number of MPI processes (QS22 nodes), where application runs.

Fig.1 part two illustrates the results for the NIPS input set. One might also observe that the execution time will be reduced if the number of MPI processes increases, similar to the KOS input dataset situation.

Reducing the execution time is due to the way application was carried out. After reading the input dataset and its distribution towards the MPI processes (this time is not taken into account), at each k-means iteration the processes look for the nearest cluster for each document belonging to it. This operation is accomplished in parallel to all MPI processes. At the end of iteration, recalculating the centers for each cluster is performed, fact that involves the information transmission amongst the MPI processes. Since this operation assumes the transmission of a small volume of information amongst the QS22 nodes, and the communications between nodes is carried out by means of InfiniBand, the loss occurred during communication are minimal. For this reason, a significant reducing of the execution time is accomplished, by the help of increasing the number of MPI processes.

7 Conclusion

In the aim of document clustering a parallel version of the k-Means algorithm was approached. This algorithm was implemented and tested on a cluster of RoadRunner hybrid architecture. Analyzing the results of the experimental tests, one might observe that by doubling the number of QS22 nodes, where the application is running, the execution time will be reduced by a percentage highly near to 50% (certain delays might occur, due to the communication amongst QS22 nodes).

The k-means algorithm is highly dependent on the first solution for cluster centers and data. In the figure 2 it may be observed that the linearization is obviously on the case when the number of processors is lower than the number of generated clusters, and the data dependency has a higher influence for a greater number of clusters.

Acknowledgement. The cluster used in order to carry out the experiments was purchased within a project entitled “Grid for Developing Pattern Recognition and Distributed Artificial Intelligence Applications–GRIDNORD”, 80/13.09.2007, PN II, Research Capacities Programme of the Romanian National Agency for Scientific Research.

References

1. Feng, Z., Zhou, B., Shen, J.: A parallel hierarchical clustering algorithm for PCs cluster system. Neurocomputing 70(4-6), 809–818 (2007)
2. Berry, M.W., Castellanos, M.: Survey of Text Mining II: Clustering, Classification, and Retrieval. Springer, Heidelberg (2008)
3. Andrews, N.O., Fox, E.A.: Recent Developments in Document Clustering (October 16, 2007)
4. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 881–892 (2002)
5. IBM BladeCenter QS22, http://www-03.ibm.com/systems/bladecenter/hardware/servers/qs22/
6. Roadrunner: Hardware and Software Overview, IBM RedBooks
7. Garcia, E.: Term Vector Theory and Keyword Weights (2005), http://www.miislita.com/term-vector/term-vector-1.html
8. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html
9. Zaharia, M.H.: Securing Communication in Ambient Networks for Speech Therapy Systems. Advances in Electrical and Computer Engineering 7(2), 41–44 (2007), doi:10.4316/AECE.2007.02010
10. Tiliute, D.E.: Security of Mobile ad-hoc Wireless Networks. A Brief Survey. Advances in Electrical and Computer Engineering 7(2), 37–40 (2007), doi:10.4316/AECE.2007.02009

Part IX
MASTS'2010 Papers

Agent-Based Information Sharing for Ambient Intelligence

Andrei Olaru and Cristian Gratie

Abstract. In a future vision of Ambient Intelligence – or AmI – our surrounding environment will integrate a pervasive, interconnected network of devices. One important concern in AmI is the exact manner in which the large quantities of generated information are managed by a system formed mostly of devices with limited capabilities. This paper presents the first steps towards the realization of the AmIciTy framework: a multi-agent system that relies on local interaction and simple context measures, having as purpose the context-aware sharing of pieces of information. A prototype has been implemented and relevant simulation results are shown.

1 Introduction

One of the priorities of current development in the ICT domain is Ambient Intelligence, or AmI [4]. Ambient Intelligence deals with assisting people in their day to day activities, by means of an integrated, ubiquitous electronic environment that is interconnected by a heavy-duty network infrastructure and providing intelligent user interfaces. AmI will use a very large number of electronic devices that have different capabilities, different sizes and different performance, all of them interconnected by wireless or wired networks, working together and cooperating towards the resolution of tasks. By means of these devices, AmI will be sensitive to the environment and to the presence and state of people, and will be able to react to their needs and actions [1, 15]. There are great challenges in the development of AmI, like advanced human-machine interfaces, knowledge representation, context-awareness, device heterogeneity, and many hardware-related requirements.

Many of these challenges relate to the layer between the hardware / network and the intelligent user interface. That is, once we have the devices, the sensors, the interconnecting network, a certain level of interoperability and the user interfaces,

Andrei Olaru · Cristian Gratie
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania
e-mail: `cs@andreiolaru.ro, cgratie@yahoo.com`

the system will generate a large amount of information, that needs to reach the users that are interested in it. Such a system must not only be context-aware, proactive and anticipative, but, as the concept of Ambient Intelligence requires, must also be reliable and fault-tolerant (so it would remain ”invisible”). These are the challenges that we try to address with the AmIciTy framework for Ambient Intelligence.

The first steps towards the realization of the framework rely on the design of AmIciTy:Mi middleware, focused on the context-aware transfer of information, in a reliable and decentralized manner, using a multi-agent system. Agents are used because they answer to the requirements of AmI: autonomy, proactivity, reasoning, reactivity [15]. The decentralized paradigm is chosen because Ami must be reliable and dependable [16], offering as many as possible of its services to the user, no matter what happens in the rest of the system, and regardless of the user's location and activity.

This paper presents the design of a multi-agent system for context-aware information sharing, in which agents interact only locally and use only very little knowledge on the environment. Context-aware behaviour is obtained by means of a few context measures that are simple and generic, yet effective for obtaining a controlled spread of information among the agents: *pressure* tells how important the information is; *specialty* tells what the information is related to (domains of interest); *persistence* tells for how much time the information is valid. Inspiration is also taken from self-organizing systems, where large numbers of individuals provide coherent global features without centralized control, relying on local interaction and feedback. This adds resilience to the system and allows for a more simple behaviour of individual agents [9].

The system has been implemented and an appropriate example application is a wireless sensor network for the tracking of events (like the passage of people or vehicles). By means of the designed context-aware behaviour, events captured by the smart sensors get to be known by the other sensors to which the events are relevant.

The next section presents some develpments the fields that this work is related to. Section 3 describes the design of the system and the measures of context-awareness, with details on the structure and behaviour of individual agents. The experimental scenarios and results are presented in Section 4. The last section draws the conclusions.

2 Related Work

The idea of creating a middleware for Ambient Intelligence is not new. In fact, most systems for ambient intelligence feature some sort of middleware, as a layer between the human-machine interface and the hardware.

Context handling is considered by the AmbieAgents infrastructure [8] in a complete manner and by means of a very well designed architecture, that also handles mobility. However, in this paper we are trying to provide a simpler agent structure, in a system that is based more on local interaction between agents. Context-aware data management is also discussed by Feng et al [5], but context queries are handled in a centralized way, making it efficient but not very scalable.

The LAICA Project [2] is a very interesting approach to an AmI middleware that shares many common points with our work. It considers an ad-hoc middleware and is based on simple agents, that provide complex functionality as an emergent feature. Nevertheless, the middleware and the agents are separate entities, and the middleware has a lower degree of distribution by using centralized operation and a central database.

Agents with reduced memory and performance footprint for AmI have been developed in the Agent Factory Micro Edition project [12]. The authors succeed in implementing a reliable communication infrastructure by using reasonably simple agents, however there is no higher level view that includes more complex global behaviour and there is no context-awareness.

The implementation of the SodaPop model [6] is another application sharing common features with our own, especially the use of self-organization for an AmI system, but it does not use the agent paradigm and it handles a quite specific case.

During the design of the middleware, we have studied many references in the domain of self-organization. Emergent properties can be of great help when dealing with limited agents but with heavy interaction. There has been much development in multi-agent systems that have emergent properties, especially by taking inspiration from biological systems [9, 10].

Self-organization has already been extensively used for wireless networks [11], taking inspiration from both biological and social (as well as other) models. Query and routing are particular features that are present in some form in our model, but in a different context and with a different final goal.

Finally, some mechanisms that we use are similar to the Directed Diffusion model [7] used for wireless sensor networks, and we are using similar techniques to spread information through the system based on local agent interaction. However, we use more context measures for better control of the spread.

This work continues our previous work on self-organizing multi-agent systems [13] and on context-aware emergent properties [14], but is more directed towards context-awareness, improving agent structure and context-aware measures, and presenting experiments that use a random placement of agents.

3 System Design

The AmIciTy middleware is being developed keeping in mind the scale and requirements of a real Ambient Intelligence scenario. In such a situation, there is a large number of users and devices (especially very many sensors and actuators). The devices communicate permanently and exchange a large quantity of information, coming from all the sensor perceptions, the users themselves, and from information aggregation. Most of the devices that are used have limited storage and processing capacity.

AmIciTy:Mi is implemented as a multi-agent system, in which agents form the middleware. New information is inserted into the middleware by inserting it into an agent, then it will spread and reach all other agents for which it may be relevant (according to context).

This is why we propose a middleware that is completely distributed. Each software agent in the middleware is assigned to and is executed on a device. Agents can communicate only if they are in a certain vicinity of each other, which is suited to our example of wireless sensor network.

Distribution and local interaction are also justified by the fact that most information is many times only relevant in a certain location or in a certain area. Decentralization also brings more reliability. From the very beginning the system was built so that it would rely on self-organization mechanisms. By means of self-organization, although generally the agents' capacities are limited and behaviour is only local, as a whole the system may perform more complicated functions and appear intelligent [2].

3.1 Context-Awareness

By context we understand the conditions in which an event occurs and that are related to the event [3]. Context-awareness is the feature of a system (here, an AmI system) that makes the system behave differently depending on these conditions.

Being context-aware means providing the user with information which is *compatible* with the context of the user. More precisely, the context of the information must be compatible with the context of the user in order for the information to be *relevant*. In our experiments, compatibility is actually between the information and the agent's context.

To provide context-awareness for information sharing, we propose four simple and generic aspects of context-awareness: first, space is implicitly considered, because of the structure of the system, that relies on local behaviour and communication; second, temporal context is implemented as a period of validity for each piece of information; third, each piece of information is related to certain domains of interest; last, each piece of information carries a direct indication of its relevance (estimated by the source).

We see context compatibility as *proximity* between the two contexts – the context of the new information and the context of the agent – in terms of the aspects enumerated above. A more detailed description of these aspects of context-awareness, together with their influence on how information is shared and spread through the system, is presented below:

Local behaviour and interaction – leads to inherent location awareness. New information will first reach the agents in the area where the information was created (e.g. where the event took place). Depending on the other aspects of context-awareness, the information will only stay in the area or will spread further. Also, all other measures equal, agents will give less relevance to information related to a farther location.

Time persistence – shows for how long the information is relevant. When its validity expires, the agents start discarding the piece of information.
Specialty – shows how the information relates to some *domains of interest*. In time, agents form their own notion of specialty depending on the information that they have. New information is considered more relevant if it is more similar to the agent's specialty, and agents share relevant information first, and they share it with agents that are more likely to consider it relevant. This influences the direction in which information is spread.
Pressure – shows how important it is for the information to spread quickly. Pressure translates into higher relevance and the agent will treat the information with higher priority. Also, the higher the pressure, the more neighbours the agent will send the information to. This way, pressure controls how quickly the information spreads.

Context compatibility, or *relevance* of new information is computed as a function of the measures of context associated with the new information and also depending on the context of the agent, comprising only an indication of specialty. In order to be able to aggregate and compare the different measures, all are quantified and bounded, and their ranges are all scaled to the interval [0, 1]: locality has an explicit quantification as the distance to the source of the event (see the next section), and is represented in the interval [0, 1], with 0 meaning that it refers to *this* agent and asymptotically growing to 1 for longer distances; persistence is a value in the interval [0, 1], with 1 meaning the information is valid forever, and 0 meaning it has expired; specialty is a vector in which each component has a value in the interval [0, 1] showing the degree of relatedness with a certain domain of interest, and the whole vector has a maximum norm of 1; pressure is also a value in the interval [0, 1].

When calculating the relevance of new information, distance, persistence and pressure are introduced directly in the computation of relevance. Specialty is compared against the specialty of the agent. Similarity between the two is calculated as follows:

$$similarity = 1 - \sqrt{\frac{\sum(S1_i - S2_i)^2}{n\ domains\ of\ interest}} sin\alpha, \quad \alpha = arccos(\frac{S1 \cdot S2}{|S1||S2|})$$

where $S1$ and $S2$ are the two specialty vectors, and the sum is for all domains of interest. The formula has been chosen in order to give lower similarity to vectors that are at greater angle (different specialties) but also to give higher similarity when one vector is less specialized than the other.

Relevance is calculated as a mean of the 4 numbers – all in the same interval – distance, persistence, pressure and similarity. This allows for different types of important facts – a fact can be equally important if it has high pressure, or if it is of great interest to the agent (similar to its specialty).

3.2 Agents

Agents in AmIciTy:Mi have been designed so that they are simple, flexible, and so that an agent with the same structure can run both on a simple processor assigned to a sensor and on a powerful computer. The agents are cognitive, but hold little knowledge and form very simple plans. In our experiments, particular attention has been given to agents that hold very small knowledge bases and that would be suited for very small devices like sensors.

In the design of the agents, inspiration was also taken from the human behaviour and thinking. As the quantity of information that will pass through an agent's knowledge base over time is quite large and the agent will be unable to (and it would probably be useless for it to) store it all, the agent must be able to sort its knowledge according to its relevance, and it must be able to "forget" information that is of no more use or of insufficient relevance.

The information in the agent's knowledge base is stored in *Facts*, where Facts are tuples of the form $\langle Agent, knows, Fact \rangle$. Note that the definition is recursive. At this point, the system is generic and does not study a real-life application. Therefore, facts that would normally represent useful information coming from the environment are replaced with Facts containing a *DataContent* placeholder, that has an identifier for tracing Facts related to that information. The *recursive depth* of a fact gives its distance to the source of the information, which is used in computing relevance.

This structure allows the agent to hold information about what it knows but also about what other agents know. This is how an agent can calculate the Specialty of neighbour agents.

In the presented experiments we have used very limited maximum sizes for the knowledge bases of agents, to show that the agents need only very little storage capacity in order to manifest context-aware behaviour. In applications where different types of devices are involved, agents may have knowledge bases of different sizes.

At the beginning of each cycle the agent checks the messages in the inbox, by integrating facts in the knowledge base, if they are new. The agent also infers that the sender knows the fact, which contributes to the agent's knowledge about its neighbours.

In the next phase the agent forms a list of potential goals. There are two types of goals that an agent can have: *Inform* other agents of some information or *Free* some storage capacity. Each goal is assigned an importance, and the most important goal will be chosen as an intention. The importance of *Inform* goals is taken from the relevance of the information to be sent. Importance for the *Free* goal is computed as a function of how full the agent's storage capacity is, reaching a value of 1 (highest importance) when the knowledge base consumes all available capacity. The agent must always have some capacity free for new facts that come from other agents.

After choosing a goal, the agent makes a plan for it. For *Free* goals, the agent decides what facts to discard. For *Inform* goals, the agent decides what neighbours to inform of the corresponding fact. The number of neighbours to inform is directly related to the pressure of the fact. Agents are chosen according to their estimated specialty, calculated as a mean specialty of the facts that the agent knows the neighbour has. At each cycle the agent will execute one action in its current plan.

The behaviour of the agent changes depending on its context measures. On the one hand, specialty directly affects the relevance that is associated with various facts. Higher relevance associated to facts makes them better candidates for *inform* messages sent to other agents, and lower relevance makes facts better candidates for removal (or ”forgetting”). Then, the agents update their own specialty according to the facts that they have.

There is an important aspect of feedback that exists in the AmIciTy:Mi . First, the way in which agents update specialty: agent’s specialty influences the facts that they consider relevant and send, sent facts influence other agent’s specialty and knowledge, and in return they receive facts with a certain specialty. Second, pressure influences relevance directly, so feedback loops form, caused by information with high pressure. Feedback leads to an aspect of self-organization by the formation of emergent specialty groups.

4 Implementation and Experiments

A proof-of-concept prototype was implemented in Java, using between 950 and 1000 agents placed in a grid or placed randomly. Experiments were focused on observing how the facts spread through the agent system. In this spreading, three things were observed: the speed with which facts spread, the coverage they reached and the particular area (or areas) into which they spread. These outputs show how the spread of information is related to the measures of context that have been assigned to it.

The types of scenarios that were studied comprised the following phases:

- insert several pieces of information into the system, with three or 4 different specialties, in order to form certain areas of interest among the agents;
- insert 3 ”test” pieces of information, of different specialties, and observe how they spread according to the previously formed areas of interest;
- insert 1 or 2 pieces of information of no particular specialty (equally related to all domains of interest) but with very high pressure, and observe if they reach all agents in the system.

In all experiments we have used specialties that relate to three domains of interest, in order be able to graphically represent the specialty as a colour having as components the components of the specialty vector. We will call these domains A, B and C.

In the first experiment that we present, 60 events are inserted into a grid of agents. The events have low pressure and persistence and one of four Specialties regarding domains A, B and C: $(1.0, 0.0, 0.0)$, $(0.0, 1.0, 1.0)$, $(0.0, 1.0, 0.0)$ and $(0.0, 0.0, 1.0)$ (let us call these Specialties A, BC, B and C). The events are inserted in one of the four quarters of the grid, according to their specialty. This allows the formation of four areas of interest, with different relatedness to the three domains of interest (see Figure 1 (b) - (d)). The four areas don’t intersect, because the facts are not of high

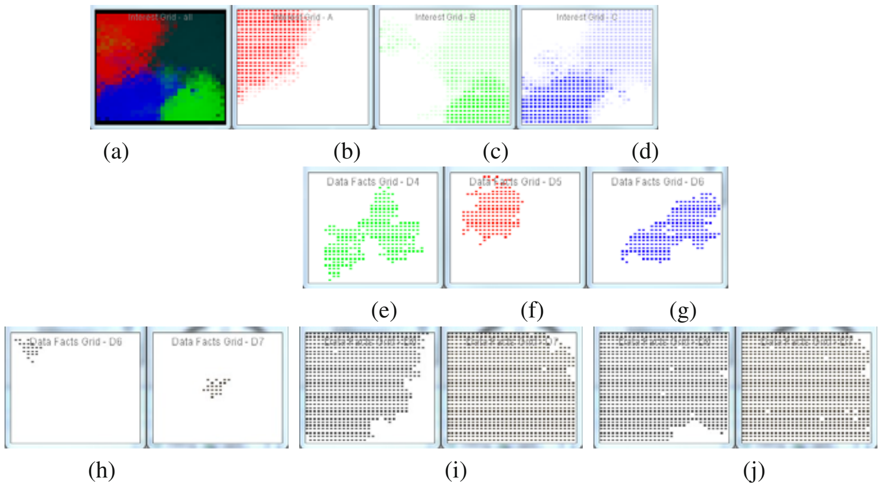


Fig. 1 (a) - (d) Areas of specialization, combined and according to domains of interest. (e) - (g) Resulting distribution of information, according to specialization. (h) - (j) evolution of the distribution of two high pressure information, starting from the corner and from the middle of the grid.

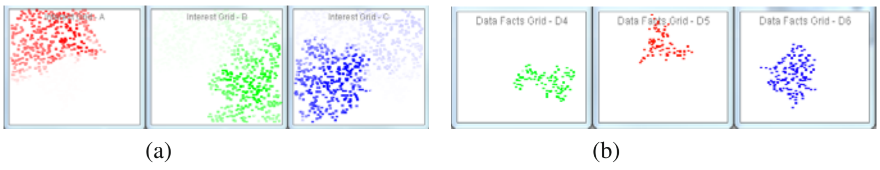


Fig. 2 (a) Areas of specialization according to the three domains of interest, with agents randomly placed. (b) Resulting distribution of data.

pressure nor very persistent, and agents that have also formed a specialty will not consider facts belonging to other specialties as very relevant (given equal pressure). This is normal, because the agents get to have the specialty of the events that are more likely to occur in the area.

Next, three new events are inserted into the center of the system, with higher pressure and persistence, and with specialties $(1.0, 0.1, 0.0)$ (call it *Ab*), $(0.0, 1.0, 0.1)$ (call it *Bc*) and $(0.1, 0.0, 1.0)$ (call it *Ca*). In the example of a smart wireless sensor network, these could be events that are truly relevant to the system, and that other sensors can take action upon, so it is of interest to sensors of the corresponding specialty to know about the event that has occurred. The resulting distribution of information is shown in Figure 1 (e) - (g). Facts of specialty *Ab* spread in the area of interest set by *A*. Facts of specialty *Ca* spread in the areas set by *C* and *BC* (domain *C*), which is normal. Finally, facts of specialty *Bc* spread in three areas: the two areas of specialties *B* and *BC* (domain *B*), but also in a part of the area of specialty *C*, because *Bc* is related to *C* as well – even if in a lower measure.

The last part of the experiment implies inserting two pieces of information with no particular specialty, but with high pressure. Observe the evolution over time in Figure 1 (h) - (j). Although the agents don't have a notion of direction (but have some little information on their neighbours), the facts spread quickly, and the fact that there are multiple facts with high pressure is not a problem. Also, as long as they still have persistence, the other pieces of information are not lost.

The results are not limited to the grid structure. Experiments have been made on agents placed randomly in the space, communicated only with agents at a distance under a certain threshold. Results obtained were of the same nature, with the observation that it took longer for information to spread, due to the many points where agents were too far to communicate. Results of the same type of experiment as above, but with randomly placed agents, are presented in Figure 2: the formed specialties (a) and the resulting distribution (b).

The results show how generic measures of context can be used, together with a simple (and fast) agent behaviour, in order to obtain context-aware behaviour. Local knowledge and simple context measures meant that knowledge bases of agents did not need to hold more than 12 root facts, among which the mean recursive depth was 2, meaning that very little memory was used.

5 Conclusion

This paper presents a multi-agent system for the context-aware sharing of information. It uses limited agents that communicate locally, together with simple, generic measures of context, in order to produce relevant results. Experiments were carried out on a large number of agents in which the different measures of context showed relevant influence on how the information spreads although agents have very limited knowledge about the system. The context measures have shown to be effective in guiding the spread of the information with respect to several parameters.

Acknowledgments. This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 1315/2008 and Grant POSDRU 5159.

References

1. Augusto, J.C., McCullagh, P.J.: Ambient intelligence: Concepts and applications. Computer Science and Information Systems (ComSIS) 4(1), 1–27 (2007)
2. Cabri, G., Ferrari, L., Leonardi, L., Zambonelli, F.: The LAICA project: Supporting ambient intelligence via agents and ad-hoc middleware. In: Proceedings of WETICE 2005, 14th IEEE International Workshops on Enabling Technologies, Linköping, Sweden, June 13-15, pp. 39–46 (2005)
3. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College (2000)

4. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.: Scenarios for ambient intelligence in 2010. Tech. rep., Office for Official Publications of the European Communities (2001)
5. Feng, L., Apers, P.M.G., Jonker, W.: Towards context-aware data management for ambient intelligence. In: Galindo, F., Takizawa, M., Traunmüller, R. (eds.) DEXA 2004. LNCS, vol. 3180, pp. 422–431. Springer, Heidelberg (2004)
6. Hellenschmidt, M.: Distributed implementation of a self-organizing appliance middleware. In: Davies, N., Kirste, T., Schumann, H. (eds.) Mobile Computing and Ambient Intelligence, IBFI, Schloss Dagstuhl, Germany. Dagstuhl Seminar Proceedings, vol. 05181, pp. 201–206. ACM, New York (2005)
7. Intanagonwiwat, C., Govindan, R., Estrin, D.: Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proceedings of MOBICOM 2000, pp. 56–67 (2000)
8. Lech, T.C., Wienhofen, L.W.M.: AmbieAgents: a scalable infrastructure for mobile and context-aware information services. In: Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005), Utrecht, The Netherlands, July 25-29, pp. 625–631 (2005)
9. Mamei, M., Vasirani, M., Zambonelli, F.: Self-organizing spatial shapes in mobile particles: The TOTA approach. In: Brueckner, S., Serugendo, G.D.M., Karageorgos, A., Nagpal, R. (eds.) ESOA 2005. LNCS (LNAI), vol. 3464, pp. 138–153. Springer, Heidelberg (2005)
10. Mano, J.P., Bourjot, C., Lopardo, G.A., Glize, P.: Bio-inspired mechanisms for artificial self-organised systems. Informatica (Slovenia): Special Issue: Hot Topics in European Agent Research II Guest Editors: Andrea Omicini 30(1), 55–62 (2006)
11. Mills, K.L.: A brief survey of self-organization in wireless sensor networks. Wireless Communications and Mobile Computing 7(1), 823–834 (2007)
12. Muldoon, C., O'Hare, G.M.P., Collier, R.W., O'Grady, M.J.: Agent factory micro edition: A framework for ambient applications. In: Alexandrov, V.N., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2006. LNCS, vol. 3993, pp. 727–734. Springer, Heidelberg (2006)
13. Olaru, A., Gratie, C., Florea, A.M.: Emergent properties for data distribution in a cognitive MAS. In: Papadopoulos, G.A., Badica, C. (eds.) Proceedings of IDC 2009, 3rd International Symposium on Intelligent Distributed Computing, Ayia Napa, Cyprus, October 13-14. Studies in Computational Intelligence, vol. 237, pp. 151–159. Springer, Heidelberg (2009) ISBN 978-3-642-03213-4
14. Olaru, A., Gratie, C., Florea, A.M.: Context-aware emergent behaviour in a MAS for information exchange. Scalable Computing: Practice and Experience - Scientific International Journal for Parallel and Distributed Computing 11(1), 33–42 (2010) ISSN 1895-1767
15. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems 23(2), 15–18 (2008)
16. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal communications 8(4), 10–17 (2001)

An Economy Model Based on Barter Exchanges between Software Agents

Andreea Urzica, Andrei-Horia Mogos, and Adina Magda Florea

Abstract. The interactions between parties within large, open systems are driven by trust. In this paper, we propose a model for barter exchanges based on reputation and trust values, which considers mechanisms for computing an agent's trust according to a set of norms enforced by a central authority. Based on this model, the paper shows how self-interested agents manage to establish cooperation relationships in order to accomplishing their goals, being aware that the costs of future transactions depend on one's reputation.

1 Introduction

The organization concept in a multiagent system has become widely accepted as a control instrument for agent autonomy, as well as a mechanism for assuring the coherent global behaviour of open systems.

Among the controlling mechanisms of a multiagent organisation, especially large scale organisation, normative multiagent systems currently represent a challenge for the research community. Norm enforcement within agent societies leads to the development of flexible organisations where agents are neither completely autonomous, as they must respect norms, nor totally restricted by a predefined behaviour.

Trust and reputation models are already known to be a key issue in designing open multiagent systems. Such models manage to synthesize the information used by an agent for the secure and efficient selection of interaction partners in uncertain situations.

This paper aims to build a connection between the normative models on one hand, and trust and reputation models on the other, in order to better simulate real-world

Andreea Urzica · Andrei-Horia Mogos · Adina Magda Florea
University Politehnica of Bucharest, Splaiul Independentei 313,
060042 Bucharest, Romania
e-mail: andreea.urzica@cs.pub.ro, mogosandrei@yahoo.com,
adina.florea@cs.pub.ro

communities. The proposed mechanism for computing trust and reputation of self-interested agents that have to establish cooperation relationships for accomplishing their goals, was inspired by the awarded board game ”Settlers of Catan”. Settlers of Catan is a multiplayer board game designed by Klaus Teuber [9]. The driving principle of this game is considered to be a relevant interaction case between agents that aim to collect various amounts of resources by barter exchange. Its mechanics is rather simple, while its dynamics is quite complex: players ”produce” resources by rolling the dice and may exchange them with other players or the bank in order to accomplish the resource combination needed for completing products.

Even nowadays there are many real-world situation where barter exchange can be found, for example, the know-how trading or service exchange. We present a scenario where agents can exchange goods only by barter and propose a mechanism that highlights the strategies that agents may use in order to maximize their profit. It uses trust and reputation concepts within the context of a normative system. The model can be adapted and applied to e-commerce applications that facilitate exchanges between collectable goods owners.

Section 2 relates our work to other papers working with the same concepts. The scenario proposed as well as the main actors are presented in Section 3. The main components of the model are described in Section 4. Some of the strategies that agents should be taking into account in order to be more efficient are addressed in Section 5. Section 6 concludes.

2 Related Work

Reputation is recognized as having both an informational role as well as a sanctioning role in social groups. In our model, placing agents in reputation classes serves only as a punitive method; it does not facilitate developing successful strategies (e.g. by choosing a lower class).

The model we describe uses a central authority enforcing the norm violation by triggering adjustments of reputation values; nevertheless each individual evaluates its partner making use of a scale of trust of its own. The Central Authority that has the power to alter one’s reputation, thus influencing its profit and its chances of success, but the responsibility of detecting a violation is left to the producer agents. Another paper supporting the idea that norm violations depend on the counterparty motivation to claim them, is [3]. The authors propose that both parties involved in a deadline obligation may have a say regarding its violation. Some other papers sustain the idea that is up to the system (e.g., an electronic organization [11] or an electronic institution [4]) to detect violations and to enforce the norms which are designed into the environment (in some cases they are even regimented in such a way that violation is not possible) [3]. The use of complaints can be also found in [2], that attempts managing trust in peer-to-peer environments. In order to allow an agent to obtain all the complaints about any other specific agent in a totally decentralized environment, the authors use a replicated storage system, presented in [1]. An important point in [2] is that it tries to deal with situations where cheating agents file groundless complaint about their partners (i.e. when their counterparts have fulfilled their engagements). In contrast with our approach, they use binary trust values, that permit no negotiation regarding contract clauses. Instead of being expelled from the community, as in [6], in our system the agents that do not conform are free to stay, but their low trust and reputation levels will prevent them from reaching the payoff regulated by the final ranking.

3 The Agent System

This multiagent community consists of agents having the same rights and responsibilities and acknowledging all the same set of norms and a Central Authority (CA hereafter), holding information about all the agents in the system (i.e. the reputation values and the production capabilities of each agent). The Central Authority periodically re-computes reputation values, publishing them on a black-board available to all agents to be consulted at no cost. Upon entering the open system, each producer agent registers to the CA and is assigned to develop a product of its choice. The product is considered to be fully developed when the agent has collected all the specified resources composing the product.An agent has a certain reputation and belongs to a corresponding reputation class, as described in Section 4.1. In addition, every agent retains a trust value for each of its partners, former or actual, computed as described in Section 4.2. Periodically, as agents dynamically enter or quit the open system, ranking results are computed considering the time consumed, the accomplished percentage from the assigned product, the reputation gained and the number of resource types it has direct access to. The rank an agent places itself determines its payoff, motivating agents to develop efficiency strategies (i.e., how to increase their production speed without losing reputation). The internal motivational engine for the agent is to achieve a place as high as possible in the periodical ranking system.

There is a set of m resource types. Each agent has direct access to a number of i types of resources ($i \in (1, m)$, different i for each agent), called *internal resources* hereafter. As there is a low probability for an agent to be able to develop its goal product using exclusively the internal resources, it has to obtain the rest of the constituent resources by trades with other agents. The resources obtained by barter with other agents, exchanged at negotiated rates, will be called hereafter *external resources*.

For each of its internal resources, an agent has assigned a specified exploiting rate. The exploiting rate shows the number of units an agent receives per tour from each resource supply.

Fig. 1 depicts the way each agent manages its internal and external resources. The darker segment shows the percentage of each acquired resource from the necessitated quantity (the entire segment). There is a set of r product types. Developing a product consists only of collecting the specified number of units from each component resource. For each pair (*product, resource type*) there is an associated amount of the *resource type* needed for the *product*.

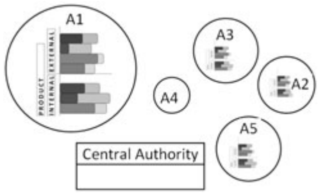


Fig. 1 Individual resource management

4 The Model

The synchronization mechanism used by the model presented here is based on tours. Each tour consists of two steps. During the first step can place either an of-fer to another agent either an inform message to the CA registering an interaction-derived norm violation. Upon receiving all the offers, the agent may accept or decline them by considering its own interests as well as the reputation information.

During the second phase, a producer agent may reply to a barter request affirmatively, implying the resource units are instantly exchanged, or by a message containing the negative answer. The CA replies to a norm violation submission with an acknowledgement message.

The list of existing agents and their production capabilities and updated reputation values is accessible to everyone, any time, at no cost.

4.1 Reputation

We consider reputation to be a global measure of agent behaviour, centralised information, available for each participant in the system, collected from multiple sources and used as a reference value. The reputation associated to an agent is represented in as an integer, proportional to the reputation level, between a minimum and a maximum value, i.e., *-RpBound* and *RpBound*. The function *Rp* associates the current corresponding reputation to each agent:

$$Rp : PA \to [-RpBound, RpBound] \cap Z, \tag{1}$$

where Z is the set of integers.

Upon entering the system every agent's reputation is initialised to 0. After a fixed number of tours (called evolving period hereafter), the Central Authority recalculates reputation values by applying the following set of rules:

```
if transactions(a) > 0 then
   if reportedNormViolations == 0 then
      if reputation(a) < RpBound then reputation(a)+=1
   else
      if reputation(a) > (-1*RpBound) then reputation(a)-=1
else reputation(a)= reputation(a)
```

where *transactions(x)* returns an integer representing the number of transactions agent x has participated in during the last *evaluation period*, and *reputation(x)* returns the reputation value agent x has achieved since entering the system.

We have chosen a discrete scale for reputation representation in order to keep the model as simple as possible but illustrative enough in the same time. There are models [5], [8], that consider the performance of an agent to be simply a bi-stable value (good or bad). We have extended the scale and introduced three classes of reputation.

Depending dynamically on the reputation values, agents are placed in three reputation classes: *Trustworthy*, *Neutral* or *Untrustworthy*, as depicted in Fig. 2 The *R* axis represents the reputation level.

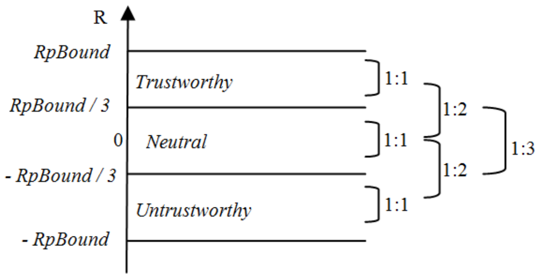


Fig. 2 Reputation classes and exchange rates

The reputation model can be easily further extended in order to take into account various functions such as diversity (lowering the weight of an opinion received too many times from the same source), the variability in the rating values [7] or validity (discarding a value that is too far away from the series of the other values received) and thus to deliver a more realistic reputation value.

Reputation can be seen as a social capital, as it influences trading interaction by altering the trust towards a potential partner, thus changing the exchange rate. The value of the same good depends on the reputation of the seller. This is what motivates agents to gain higher reputation values by respecting their contracts.

4.2 Trust

In this paper trust is considered to be the subjective opinion of an agent upon the behaviour of its trading partners, based on direct interactions.

The value representing how much agent *A1* trusts agent *A2* is represented also as an integer and uses the same grid as the global reputation view. Each agent keeps an updated table of trust values for all the agents it has interacted with. The simulation begins with all agents having the trust table empty. Upon deciding whether to exchange resources with an unknown agent, an agent uses the publicly accessible reputation value of the new agent (provided by the CA) as trust value.

After a fixed number of tours (the *evaluation period*), every agent re-computes the trust with respect to every other agent it has ever collaborated with by applying the following set of rules:

```
if interactions(a1,a2) > 0 then
   if normViolation == 0 then
      if trust(a1,a2) < RpBound then trust(a1,a2)+=1
   else
      if trust(a1,a2) > (-1*RpBound) then trust(a1,a2)-=1
else trust(a1,a2)= trust(a1,a2)
```

where *interaction(x,y)* returns an integer representing the number of interactions between agents x and y during the last *evaluation period*, and *trust(x,y)* returns the trust value agent x has with respect to agent y.

4.3 Norms

We consider the set of norms proposed by this model to be ”impositive”, i.e., norms forcing an entity to do an action or to reach a state, according to the classification found in [10]. In order to simplify scenario we have chosen only one general simple rule: Agents must respect the exchange rates shown in Fig. 2, according to the reputation class it belongs to. The exchange rate between an agent belonging to the *Trustworthy* class and an agent belonging to the *Untrustworthy* class is 1:3 (for each unit the agent belonging to the superior class gives it receives 3 units), 1:2 for agents belonging to adjacent classes and 1:1 between agents belonging to the same class.

Both the Central Authority and the producer agent are responsible for norm enforcement. An agent can place a complaint to the CA whenever a contract failure occurs, i.e. upon an exchange the agent was not provided the promised number of units of the considered type. An agent can observe a norm violation only if it has occurred during one of its interactions with other agents.

4.4 Implementation Results

Based on different parameters that evaluate the accomplishments of each agent during the simulation, ranking results are computed periodically. The payoff the agent pursues is inverse proportional with the rank it achieves. The parameters considered in measuring the fitness of an agent may be: the time consumed developing the assigned product, the final reputation value and the initial assets (how many of the internal resource types were useful in developing the assigned product).

We have developed a testbed in order to tune the input of the system by various parameters: the number of agents, of resources, of products, product ingredients and quantities, resources production rates for each agent as well as various thresholds and probability values that influence the data generator or the agent behaviour. We have run multiple experiments with sets from 40 to 100 agents. We have varied the evaluation period from 100 rounds to 20 rounds and we have observed that the

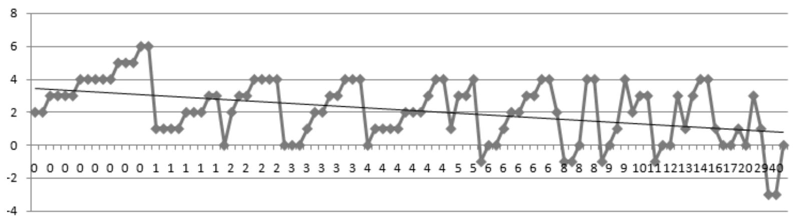


Fig. 3 Reputation values depending on norm violations

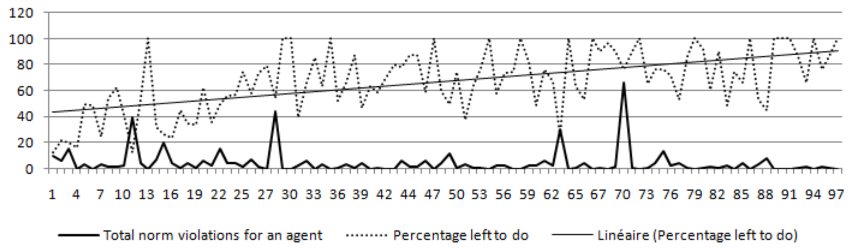


Fig. 4 Norm violations and initial assets depending on time ranking

smaller the evaluation period is, the greater the reputation values are, as the agents have more opportunities to correct their reputation and they have fewer chances to violate a norm during that period. An example of the evolution of reputation values along with the total number of norm violation per agent is presented in Fig. 3.

In Fig. 4 agents are lined up along the abscissa by their place in the time ranking (the agent identified by 1 was the first that finished its product). The chart presents how do the norm violations and the internal resources influence the speed of developing the product.

Percentage left to do constitutes the amount of external resources that have to be collected. The tendency curve shows that agents having more resources left to collect spent more time in fulfilling the product. By analyzing the norm violations distribution we observe small number of very untrustworthy agents. The overall system can be considered well balanced as these very untrustworthy agents, al-though they rarely occupy the last places, they never reach the first places.

5 Strategies

Given the set of norms, agents should develop and use a set of strategies for selecting the interaction partners. An agent may decide to inform the Central Authority upon observing a norm violation or not. The action resulting from this decision process may vary, however, upon the progress of its goal product. As the communication with the CA costs one tour, an agent may choose not to report the fraud. The producer agent will be interested into informing the CA upon a norm violation when in the early stage of developing its goal product, as the penalties received by other agents lower their reputation and places them into inferior classes. The transactions with agents belonging to an inferior class are more profitable even though they imply a higher risk degree. When in its late stage of realizing the product, the agent's interest in informing the Central Authority of a norm violation may decrease as it tries to complete its goal as soon as possible and may not afford to waste a tour. Thus, its contribution to the accuracy of the global reputation system diminishes towards the final stages of development of its own product. These are decisions internally managed by each individual. Upon finishing its product, the agent's interest in informing the CA of a norm violation may increase as it is in its favour that all the other agents finish their products as late as possible; thus, the agent will make a complaint every time it observes a fraud.

An agent violates a norm when not providing all the promised units of resource in conformance with the agreed contract. We tend to think that this behaviour appears more frequently when an agent has almost completed its product, and therefore, it values time more than reputation. Given the synchronization mechanism provided by the proposed model, we consider that this sort of norm violation are made by choice, as agents can evaluate all offers, thus knowing where they are going to invest their resources for the current tour, before agreeing upon a deal. When evaluating the offers received or when inspecting the list of the agents offering the needed resource type, an agent *A1* computes the reputation class for each of these agents using the following two rules:

1. If there is no previous collaboration between the two agents, the evaluator, A1, considers its trust in A2 is directly equal to the reputation of A2, provided by the Central Authority.
2. If the evaluator agent, A1, already has a record of the evaluated agent, A2, (i.e. they have collaborated before), then the reputation class of A2 is given by the average between the publicly available reputation of A2 and the trust agent A1 has with respect to A2.

The first rule expresses the influence of the acquired reputation in initialising a relationship. The second rule shows that the personal experience influences the perception upon one's reputation. The motivation for the second rule comes from the need to compensate the possible anomalies in computing the global reputation. There may be cases when a malicious agent, agent D, intends to denigrate the reputation of the norm-compliant agent A, and send fake complaints to the CA about agent A. If agent B had previous interactions with agent A and its internal trust value for agent A is high, the use of the average between its trust value and the damaged reputation will bring it closer to the real value.

All agents would be interested in lowering their peer's position on the reputation grid, even when the public reputation is high, as they can lie about their internal trust value about their peer. This situation is prevented as a contract cannot be established until both parties agree upon the reputation class where they have been placed by the interaction partner. This is why negotiation is of major importance. The public-posted reputation level can always be used as a reference value during negotiations and it is in both parties interest trying to place its partner on a position as accurate as possible against the reputation grid, in order to come to an agreement.

For further development, the level each agent places its partner on the reputation grid would be more realistic determined by using a weighted mean. The coefficients used by the weighed mean are related to the number of previous interactions, and inverse proportional to each other. This way, agent A would give more importance to agent B's reputation as it had little or no interaction with B, and will act more accordingly to its own personal rating on agent B's trustfulness as they have more past interaction episodes.

6 Conclusions

This paper proposes a model for a multiagent system that combines the use of trust and reputation together with the enforcement of a set of norms.

The model uses non-monetary trading interaction. The trading partner selection process considers both the peer's achieved reputation (a global measure) and the subjective trust value (set by previous direct interactions). The trust values agents assign to each other is computed according to the set of norms featured by the system. The model addresses the penalty issue by placing agents in different reputation classes, a different status implying different costs for the same goods.

The paper studies the way norms are enforced by the use of a central authority. Since the Central Authority solution has two important drawbacks, namely, it overcharges the communication network and represents a single point of failure, we are further studying ways of implementing a decentralised deception-proof solution. As future work we should also investigate how to deal with the possibility of an agent placing a complaint upon a dishonest review and implement a second type of reputation: evaluator reputation.

Acknowledgements. This research was supported by Grant CNCSIS ID 1315, 2009-2011, and Grant POSDRU ID 7713.

References

1. Aberer, K.: P-Grid: A self-organizing access structure for P2P information systems. In: Cooperative Information Systems, pp. 179–194. Springer, Heidelberg (2001)
2. Aberer, K., Despotovic, Z.: Managing trust in a peer-2-peer information system. In: Proceedings of the tenth international conference on Information and knowledge management, p. 317. ACM, New York (2001)
3. Cardoso, H., Oliveira, E.: Directed Deadline Obligations in Agent-based Business Contracts. In: Proceedings of the international workshop on Coordination, Organization, Institutions and Norms in agent systems, COIN@ AAMAS 2009 (2009)

4. Esteva, M., Rodriguez-Aguilar, J., Sierra, C., Garcia, P., Arcos, J.: On the formal specification of electronic institutions. In: Agent mediated electronic commerce, pp. 126–147 (2001)
5. Mui, L., Mohtashemi, M., Halberstadt, A.: A Computational Model of Trust and Reputation for E-businesses. In: HICSS, p. 188. IEEE Computer Society, Los Alamitos (2002)
6. de Pinninck, A., Sierra, C., Schorlemmer, M.: Friends no more: norm enforcement in multiagent systems. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, p. 92. ACM, New York (2007)
7. Sabater, J., Sierra, C.: REGRET: reputation in gregarious societies. In: Proceedings of the fifth international conference on Autonomous agents, pp. 194–195. ACM, New York (2001)
8. Sen, S., Sajja, N.: Robustness of reputation-based trust: Boolean case. In: Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1, p. 293. ACM, New York (2002)
9. Site, C.O., http://www.catan.com/en/download/?SoC_rv_Rules_091907.pdf (last accessed 2009)
10. Vázquez-Salceda, J., Aldewereld, H., Dignum, F.: Implementing norms in multiagent systems. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 313–327. Springer, Heidelberg (2004)
11. Vázquez-Salceda, J., Dignum, F.: Modelling electronic organizations. In: Mařík, V., Müller, J.P., Pěchouček, M. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 1070–1070. Springer, Heidelberg (2003)

Finding Network Resources by Using Mobile Agents

Gabriel Ciobanu

Abstract. This paper describes a distributed software platform capable of hosting agents whose purpose is to offer and to find effective resources having optimal utility. Brokers are agents offering the resources; the explorers are agents moving along the network according to an algorithm which takes into account the latency (of the migration channel) and the CPU load (of the destination) to ensures an optimal exploration.

1 Introduction

In general, agents provide a way to describe complex systems by entities capable of acting with a certain degree of autonomy in order to accomplish tasks. The fundamental idea of an architecture for mobile agents is that it identifies components and their interconnections. We use a software platform for mobile agents with timers allowing timed migration, local interactions, and mobility between explicit locations. The aim of the paper is to describe a resource discovery in a distributed environment using mobile agents able to find effective resources having optimal utility.

We start by providing the syntax and semantics of the mobile agents. They are described as a process calculus, and use timing constraints to specify what is the time window for a mobile process to move to another location. The process calculus is mainly based on TiMo, a formalism using timeouts for interactions and adaptable migration between explicit locations [2]. An implementation corresponds rigorously to the semantics of TiMo is presented in [1]. A further step relating migration to resources is given by the distributed

Gabriel Ciobanu
Institute of Computer Science, Romanian Academy, Iaşi
and "A.I.Cuza" University of Iaşi, Romania
e-mail: `gabriel@iit.tuiasi.ro, gabriel@info.uaic.ro`

π-calculus [4] which have introduced process migration and access control. A distributed π-calculus with timers is formally presented in [3].

We assume that $Chan$ is a set of channels, Loc is a set of locations, Var is a set of location variables, and $Ident$ is a finite set of process identifiers (each identifier $id \in Id$ has a fixed arity $m_{id} \geq 0$). Communication on a channel or a movement to new location is no longer indefinite; if an action does not happen before a predefined deadline, the waiting process switches its operation to an alternate mode. This approach leads to a method of sharing of the channels over time. A timer (such as $\Delta 7$) of an output action $a^{\Delta 7}\,!$ makes it available for communication only for the period of 7 time units. We use timers for both input and output actions. The reason for having the latter stems from the fact that in a distributed system there are both multiple clients and multiple servers, and so clients may decide to switch from one server to another depending on the waiting time. The syntax of the mobile agents is

$$\begin{array}{llll}
\textit{Processes} & P, Q & ::= a^{\Delta t}\,!\,\langle v\rangle\,\texttt{then}\,P\,\texttt{else}\,Q \;\mid & \text{(output)}\\
& & \quad a^{\Delta t}\,?\,(u)\,\texttt{then}\,P\,\texttt{else}\,Q \;\mid & \text{(input)}\\
& & \quad \mathbf{go}^{\Delta t}\,v\,\texttt{then}\,P \;\mid & \text{(move)}\\
& & \quad \mathsf{stop} \;\mid & \text{(termination)}\\
& & \quad id(v) \;\mid & \text{(recursion)}\\
& & \quad P\,|\,Q \;\mid & \text{(parallel)}\\
& & \quad \circledS\, P & \text{(stalling)}
\end{array}$$

$$\textit{Networks} \quad M, N ::= l[\![P]\!] \;\mid\; M\,|\,N$$

In the above description, it is assumed that $a \in Chan$ is a channel, and $t \in \mathbb{N} \cup \{\infty\}$ is a time deadline, v is a location constant or variable, u is a location variable, and $\circledS$ is a special symbol used to express that a process is temporarily stalled. For each process identifier $id \in Id$, there is a unique definition $id(u_1, \ldots, u_{m_{id}}) = P_{id}$, where $u_i \neq u_j$ (for $i \neq j$) are variables acting here as parameters.

Mobility is implemented using processes like $\mathbf{go}^{\Delta t}\,u\,\texttt{then}\,P$. It can move within t time units to the location u and behave as process P. Note that since we allow v in the $\mathbf{go}^{\Delta t}\,v\,\texttt{then}\,P$ construct to be a variable and so its value is assigned dynamically though communication with other processes, migration actions support a flexible scheme for movement of processes from one location to another.

The output process $a^{\Delta t}\,!\,\langle v\rangle\,\texttt{then}\,P\,\texttt{else}\,Q$ attempts to send the location given by v over the channel a for t time units. If this happens, then it continues as process P, and otherwise it continues as the alternate process Q. The input process $a^{\Delta t}\,?\,(u)\,\texttt{then}\,P\,\texttt{else}\,Q$ attempts for t time units to input a location and substitute it for the variable u within its body.

Processes are further constructed from the (terminated) process stop and parallel composition $(P|Q)$. A located process $l[\![P]\!]$ is a process running at location l. Finally, process expressions of the form Ⓢ P represent a purely technical notation which is used in our formalization of structural operational semantics of TiMo; intuitively, it specifies a process P which is temporarily blocked and so cannot execute any action.

The operational semantics of the mobile agents is presented in [2]. It is based on two kinds of transition rules: $M \xrightarrow{\lambda} N$ and $M \xrightarrow{\surd} N$, where the former is recording an execution of an action λ, and the latter is a time step. The action λ can be $k \triangleright l$ or $a\langle l \rangle @k$ or $@k$, where k is the location where the action is executed and l is either the location a process has moved to, or the location name communicated between two processes over the channel a. Moreover, $@k$ indicates that a process which could have moved to another location decided to wait in the same location k for another time unit.

2 Network Resource Discovery Platform

We use an abstract notion of resource able to describe both physical and symbolic features. A resource is given by a set of pairs (*action*, *characteristic*), where characteristic is presented as *type: value*. Actions and characteristics together distinguish among the existing resources: the *characteristics* define the required features of the available resources allowing the specific *actions*. Our new feature is given by the use of a migration platform and mobile agents to discover the resources. This approach by migration enables the agents to interact with the hosts and take context-aware actions rather than just query the resources as in RDF. For instance, someone can send an agent to effectively discover a working color printer with a high resolution; after it finds such a printer, the agent can actually use it and then report the location of the printed document (by a message "your document is printed in office B203").

The resource discovery platform is build using a framework called MCTools, namely a framework for agent migration with a multi-layered model. MCTools framework allows the agents to roam and travel freely in a network of machines, being developed according to a choreography based distributed architecture, i.e. working without a central coordinator. The agents of the resource discovery platform are aimed to detect soft and hard resources with specific features and with an optimal utility. The general framework serves as a base for implementation of various platforms for mobility, dealing with names, locations, agents, migration, resources etc. It provides a default mechanism for migration which makes possible to migrate an agent by its code and data. It also handles communications with other machines, and thus it can create and initialize a distributed environment from a global specification, making it a useful tool for distributed experiments. The framework is based on an extensible software architecture such that the majority of components can be customized. It is implemented in Java language, the main reason being the infrastructure offered by Java for working with mobile code and dynamic classes. More details on the general framework architecture are presented in [1].

We use a module called Resource Model providing a representation for the network resources (actions, characteristics). This module contains the needed structures to express the resources constraints (pairs of actions and characteristics). It works with some abstract representations which are implemented by specific resources. An Execution Model is defined by primitives such as migration and communication; these primitives can express other processes such as starting or stopping the execution, or joining with other execution threads. It defines both the explorer (looking for the resources) and the broker (offering the resources). Execution model runs inside a specific Execution Environment. Some basic functionalities of an execution environment are adding or removing new agents, and generating unique names. It is worth noting that there are many functionalities which are transparent to the developer; e.g. communication with other machines, not just for transmitting agents, but also for synchronization and control. The execution environment contains the Resource Manager which intermediates between the explorers and brokers. The brokers register their resources to the Resource Manager together with an access protocol. The explorers interact with the resource manager which search for the required resource into the brokers repositories. The specific access protocol of each broker is applied in order to get the resource.

The execution environment provides also a safe bridge to host in order to allow the agents to interact properly with the physical available resources. We have a network discovery module which find the neighbours in terms of IP address and ports. For each neighbour, the latency and CPU load is registered. This allow to select the minimum latency for migration (here the resource is the migration channel), and to avoid moving agents on busy hosts. On the other hand, a broker can decide the migration to another host whenever the current host becomes too busy, or it exists a requirement for such a movement. Working with interfaces, each resource can implement its specific characteristics and behaviour. This means that it is possible to allow resources with better characteristics in response to a certain query (for instance, finding a printer with a speed of 22ppm is a valid response for an agent looking for a printer with a speed of 15ppm).

Regarding the use of the platform, the resources are defined by extending the framework base classes to describe their features. The actions and characteristics are derived from their corresponding abstract classes. We provide a generic implementation for the explorer and the broker, and the user can extend them according to specific requirements. Other custom agents can be easily implemented in Java, according to MCTools user indications [1].

3 Discovery Process and Optimal Utility

The aim of the resource discovery platform is to support the mobile agents looking for certain soft and hard network resources having a maximum utility. Each location has a Resource Manager and one or more resource brokers. The resources have some characteristics. The resource explorers migrate in the network and look for the set of resources with an optimal utility; they keep the their locations, and the paths to them. The agents build dynamically the path to a resource, selecting the next host based on the current latencies and the CPU loads of the neighbours. The agent discovering process is adaptive, and it could be described by an algorithm which takes into account several network quantitative aspects to ensures an optimal exploration.

Therefore the resource discovery platform has three main actors:

- **The agent** explores the network in search of a specific resources.
- **The broker** is installed on a specific location and offer a set of resources. It can move and install to another location. It can impose a set of condition which agents have to satisfy in order to access the resources.
- **The resource manager** is unique on each location, and has the responsibility to authenticate agents and brokers, acting as a safe bridge to the host resources.

We use a simple scenario to point out some key functionality of the platform. First, an agent is designed to find a certain resource with specific characteristics. It is the loaded in the platform at is home location where it starts its search. If it cannot find a resource with utility 1 at the current location it stores the one with the maximum utility and requests the next location for migration. The next destination is computed according to the latency and CPU load of all neighbours. Once that a new location is set the agent migrates and starts over. It finishes when it finds a resource with utility 1 or when it has explore all the network. Finally it returns to the owner and report the resource with the maximum utility, its location and the path to get there.

Another interesting feature of the platform is that brokers may also move between locations. In this way even the broker can optimize the system by moving to a better host (better connection speed and hardware performance). However this is not always possible because the broker resources might be bind to its host.

Security could be handled at location level by a *Location Manager*, and at the resource level by broker. An agent may find the resource it looks for, but not having permissions to retrieve it. In this case it duplicates itself, and one copy continue to search while the other return to the owner to inform about the situation.

The explorers search resources based on characteristics constraints. Such constraint is expressed in form of $(action, \psi)$ where ψ is a function defined over the set of characteristics and returns whether or not the specified characteristic satisfies the constraint. The set of all characteristics constraints for a desire resource is named the resource constraint.

The resource utility is mainly computed by dividing the number of satisfied constraints with the total number of constraints. In fact, it is possible to indicate a subset of resource constraints which are absolutely necessary. Utility is 0 is this subset is not satisfied. For instance, we can require 10 constraints for an optimal resource discovery, among which 3 are essential. If the explorer finds resources satisfying only 7 constraints (including those 3 essential constraints), then the utility is 0.7. However if not all the 3 essential constraints are among them, the utility is 0.

Formally the resource model is defined as follows. Let A be the set of actions, and C the set of characteristics. A resource R is a set $\{(a,c) \mid a \in A, c \in C\}$, while a resource constraints RC is a set $\{(a,\psi) \mid a \in A\}$, where

$$\psi : C \to \{0,1\} \quad \psi(c) = \begin{cases} 1 \text{ if } c \text{ satisfies the constraint} \\ 0 \qquad\qquad \text{otherwise} \end{cases}$$

If R is a resource, its utility with respect to the resource constraint RC is

$$\mathit{utility}(r, rc) = \tfrac{1}{|rc|} \cdot \textstyle\sum_{(a,c)\in R} \sum_{(a',\psi)\in RC} \iota(a,a') \cdot \psi(c), \quad \text{where}$$

$$\iota : A \times A \to \{0,1\} \quad \iota(a,a') = \begin{cases} 1 \text{ if } a = a' \\ 0 \text{ otherwise} \end{cases}$$

This resource utility can be extended to support weighted constraints.

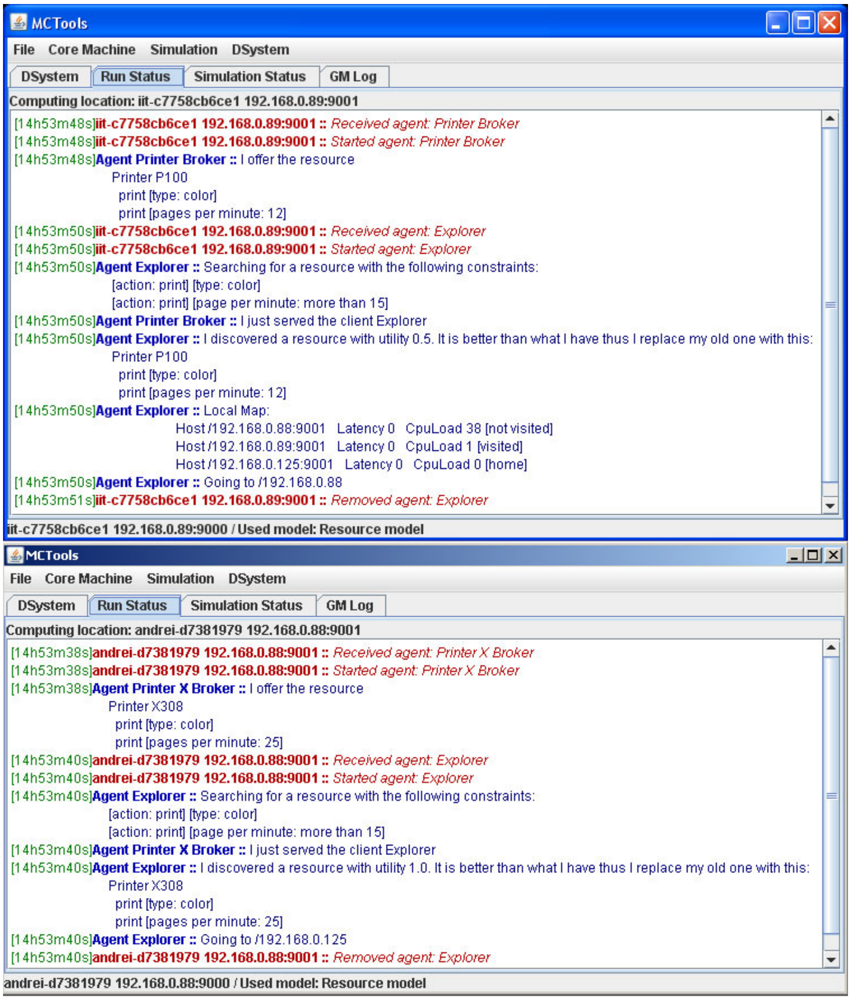
At the implementation level, ψ is an abstract function which takes as argument a characteristic and returns a boolean value. It must be implemented in all the custom constraints. A resource constraint is also represented by an object with a list of characteristic constraints. This object computes the utility of a resource. It has a final method which takes as argument a resource, and return its utility with respect to the constraints.

When a request arrives at the Resource Manager, this iterates trough all resources, and present them to the resource constraint object which then computes their utilities. The resource with the maximum utility is returned.

Table 1 Network Resources

	action	characteristic
Scanner S205	scan	type: color
	scan	dpi resolution: 600
Printer P100	print	type: color
	print	pages per minute: 12
Printer X308	print	type: color
	print	pages per minute: 25

We give a simple example which illustrate the discovery process. The example scenario is determined by the finding of a color printer capable of printing more than 15 pages per minute in a local network. We assume that in the network are available the following resources: a color scanner with the resolution of 600dpi, a color printer with speed of 12 pages per minute and one with 25 pages per minute. This scenario can be translated into the resource discovery platform as follows. An explorer is sent to find a resource with maximum utility satisfying the following constraints: (print, [type:color]),


MCTools
File Core Machine Simulation DSystem
DSystem Run Status Simulation Status GM Log
MCTools
File Core Machine Simulation DSystem
DSystem Run Status Simulation Status GM Log

```
Computing location: iit-c7758cb6ce1 192.168.0.89:9001
[14h53m48s]iit-c7758cb6ce1 192.168.0.89:9001 :: Received agent: Printer Broker
[14h53m48s]iit-c7758cb6ce1 192.168.0.89:9001 :: Started agent: Printer Broker
[14h53m48s]Agent Printer Broker :: I offer the resource
                   Printer P100
                     print [type: color]
                     print [pages per minute: 12]
[14h53m50s]iit-c7758cb6ce1 192.168.0.89:9001 :: Received agent: Explorer
[14h53m50s]iit-c7758cb6ce1 192.168.0.89:9001 :: Started agent: Explorer
[14h53m50s]Agent Explorer :: Searching for a resource with the following constraints:
                   [action: print] [type: color]
                   [action: print] [page per minute: more than 15]
[14h53m50s]Agent Printer Broker :: I just served the client Explorer
[14h53m50s]Agent Explorer :: I discovered a resource with utility 0.5. It is better than what I have thus I replace my old one with this:
                   Printer P100
                     print [type: color]
                     print [pages per minute: 12]
[14h53m50s]Agent Explorer :: Local Map:
                        Host /192.168.0.88:9001   Latency 0   CpuLoad 38 [not visited]
                        Host /192.168.0.89:9001   Latency 0   CpuLoad 1 [visited]
                        Host /192.168.0.125:9001   Latency 0   CpuLoad 0 [home]
[14h53m50s]Agent Explorer :: Going to /192.168.0.88
[14h53m51s]iit-c7758cb6ce1 192.168.0.89:9001 :: Removed agent: Explorer
iit-c7758cb6ce1 192.168.0.89:9000 / Used model: Resource model
```

```
Computing location: andrei-d7381979 192.168.0.88:9001
[14h53m38s]andrei-d7381979 192.168.0.88:9001 :: Received agent: Printer X Broker
[14h53m38s]andrei-d7381979 192.168.0.88:9001 :: Started agent: Printer X Broker
[14h53m38s]Agent Printer X Broker :: I offer the resource
                   Printer X308
                     print [type: color]
                     print [pages per minute: 25]
[14h53m40s]andrei-d7381979 192.168.0.88:9001 :: Received agent: Explorer
[14h53m40s]andrei-d7381979 192.168.0.88:9001 :: Started agent: Explorer
[14h53m40s]Agent Explorer :: Searching for a resource with the following constraints:
                   [action: print] [type: color]
                   [action: print] [page per minute: more than 15]
[14h53m40s]Agent Printer X Broker :: I just served the client Explorer
[14h53m40s]Agent Explorer :: I discovered a resource with utility 1.0. It is better than what I have thus I replace my old one with this:
                   Printer X308
                     print [type: color]
                     print [pages per minute: 25]
[14h53m40s]Agent Explorer :: Going to /192.168.0.125
[14h53m40s]andrei-d7381979 192.168.0.88:9001 :: Removed agent: Explorer
andrei-d7381979 192.168.0.88:9000 / Used model: Resource model
```

Fig. 1 Simple and Optimal Printer Locations

(print, [pager_per_minute: more than 15]. The existing resources are offered by different brokers and are expressed as in Table 1.

In order to be able to compare numeric characteristics values, a custom constraint was implemented. The running of a this scenario can be observed in Figures 1; the details are presented below.

The Explorer search first the local host, but it finds only the scanner which has no utility since it has no *(action, characteristic)* that matches the constrains. Then he queries for local neighbours completing its local map with the corresponding latencies and CPUs loads. To decide where to go next the agent computes for each unvisited neighbour a score based on the latency and CPU load. The destination is the neighbour with the best score. This provides the agent with the ability to avoid network failures and to adapt to the network context. For this example we take the score formula as $score = latency + CpuImportance * CpuLoad$ where $CpuImportance$ was set to 0.1. Thus the best score is actually the lowest score.

Based on this formula the agent decides to migrate at address 192.168.0.89 (upper Figure 1). Here the printer broker offers access to a color printer but with only 12 pages per minute. The utility of such resource is only of 0.5. Since no other resource was found yet, *the explorer* decides to stores it. Because the current utility (0.5) is lower than the needed one(1.0), *the explorer* request the neighbours again and than moves to 192.168.0.88 (lower Figure 1) based on the same choosing algorithm. At 192.168.0.88 *the explorer* finally finds a resource with utility 1, meaning that all the resources constrains are satisfied. Because it has found a resource with utility equal with the needed one, *the explorer* returns home and report the result and the time in which the task was completed.

4 Conclusion

The paper describes the use of mobile agents to do resource discovery in networks. We focus on using utility measures to optimize selection among available resources. The problem of resource discovery in networked environments has been researched during the last years. However in this paper we work with mobile agents derived from a process calculus, and so having a formal semantics and several theoretical results that could facilitate development and verification. We describe a migration platform used for network resource discovery. Using an abstract notion of resource, we use brokers and explorers to find effective resources having optimal utility. Both the brokers and explorers have context-aware adaption behaviours based on the current network status. For instance, brokers migrate to another host whenever the current one gets busy, and the explorers adapt their search based on the network latency and destination CPU load. From the examples we present results that CPU load is not very significant; the model can add new possible resource features.

Acknowledgement. I am very grateful to Călin Juravle and Maciej Koutny for their contribution and fruitful collaboration.

References

1. Ciobanu, G., Juravle, C.: A Software Platform for Timed Mobility and Timed Interaction. In: Lee, D., Lopes, A., Poetzsch-Heffter, A. (eds.) FMOODS 2009. LNCS, vol. 5522, pp. 106–121. Springer, Heidelberg (2009)
2. Ciobanu, G., Koutny, M.: Modelling and Verification of Timed Interaction and Migration. In: Fiadeiro, J.L., Inverardi, P. (eds.) FASE 2008. LNCS, vol. 4961, pp. 215–229. Springer, Heidelberg (2008)
3. Ciobanu, G., Prisacariu, C.: Timers for Distributed Systems. Electronic Notes in Theoretical Computer Science, vol. 164, pp. 81–99. Elsevier, Amsterdam (2006)
4. Hennessy, M., Riely, J.: Resource Access Control in Systems of Mobile Agents. Information and Computation 173, 82–120 (2002)

VICAL: Visual Cognitive Architecture to Understanding Image Semantic Content

Yamina Mohamed Ben Ali

Abstract. In this paper, we are interested by the different sides of visual machine learning. For this purpose, we present an expected cognitive architecture to highlight all the visual learning functionalities. Despite the fact that our investigations were based on the conception of a cognitive processor, as a high interpreter of object recognition tasks, we strongly emphasize on a novel evolutionary pyramidal learning. Indeed, this elaborated approach based on association rules enables to learn highest concepts in order to understand the highest semantic content of an input image.

1 Introduction

Visual learning is an attractive trend in object recognition investigation because it seems to be the only way to build vision systems with the ability to understand a broad class of images. Indeed, visual learning is a complex task that usually requires problem decomposition, big amount of data processing, and eventually an expensive time consuming process. In most approaches to visual learning reported in the literature, learning is limited to parameter optimization that usually concerns a particular processing step, such an image segmentation, feature extraction, etc. Only, a few contribution attempt to close the feedback of the learning process at the highest level (i.e. recognition) [4],[12],[13],[14],[15],[16],[17],[19],[21].

In this paper, we are interested by the different sides of the visual machine learning. For this purpose, we have focused our investigations on a novel cognitive architecture to outperform all visual tasks in the domain of object recognition, as well as the semantic content of a given image. Section 2 describes the content of a

Yamina Mohamed Ben Ali
Computer Science Department,University of Badji Mokhtar,
BP 12, 23000, Annaba, Algeria
e-mail: `Benaliyam2@yahoo.fr`

visual cognitive processor. Section 3 illustrated the structural abstraction of VICAL. Finally, we conclude the paper by some perspectives.

2 Visual Cognitive Architecture

The visual cognitive behavior represents one of the main properties of the cognitive processor. Described at the meta-learning level, the cognitive processor embodies some conceptual modules relative to the interpretation of knowledge provided from sensorial organs. Indeed, besides the fact that such a processor has the hard charge of supervising, communicating, and distributing processing tasks, it is able to solve in parallel some cognitive tasks like visual recognition, speech recognition etc.

In our studies, we are interested by the visual cognitive task which requires the introduction of two behavioral components: the visual memory and the visual module agent. On the first hand, we consider the visual memory as a visual repository including information represented as concepts. The term of concept denotes an elementary semantic content like eye, nose, arm, etc. or a composed semantic content like face, human, cat, house, etc. Thus, the information coming from the environment, after image segmentation, consists of geometrical descriptors indexed as elementary forms or concepts. On the second hand, and as we see in Fig. 1, the visual module represents the agent that supervises all learning transactions done by either the operational agent or the evolutionary agent.

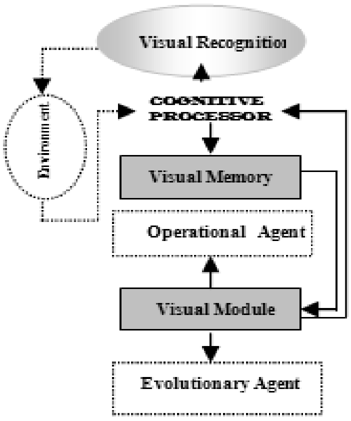


Fig. 1 Cognitive visual architecture

The main activities of the visual agent are mentioned by the following steps:

1. Select initially formal concepts of rank k (k=0)
2. Locate the appropriate database (used for learning concepts) according to the similarity measure between the given concepts and the keywords used for indexing this database.

3. In case where such matching is possible, apply the evolutionary procedure to learn higher concepts and go to step 4. Otherwise, the final formal concepts are proposed as the semantic content of the processed image.
4. Replace the formal concepts of rank by the new learned concepts of rank k+1.
5. Compute distances between new concepts.
6. Compare these distances with an empirical threshold (build the new sets of concepts).
7. Go to step 2.

3 Visual Agent Module

This section describes the organizational structure of VICAL. Based upon the fact that the recognition process is strongly influenced by two cognitive properties, namely evolutionary concept learning and clustering objects, the suitable organization is a role-guided structure. Thus, the primary task of the visual agent is to ensure the communication between the operational agent and the evolutionary agent. The aim is to search and retrieve concepts from a lower level to a higher level during the learning process, as shown in Fig. 2.

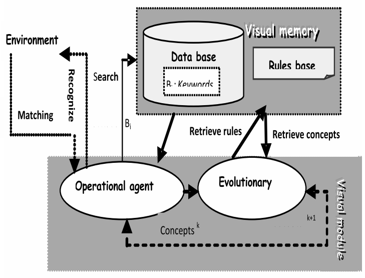


Fig. 2 Diagram of interactions between agents for solving visual problem

3.1 *Operational Agent*

The operation agent focuses on the geometrical aspect of a concept rather than on its semantic aspect. Thus, it manipulates objects regarded as vectors of integer coordinates, and carries out its clustering role according to the position of objects in the image. Let us assume that the number of salient objects detected in the image (before we lunch the learning algorithm) is n and the set of objects is $O = \{O_1, O_2, \ldots, O_n\}$. The operational agent constructs the matrix of correspondence $M(nxn)$ where the rows and the columns are the objects. The construction of the matrix is done according to the measure of the distance between pairs of objects. Thus, since the objects represent edges or contours we assume that the appropriate distance is the Hausdorff distance which computes the maximum distance of a set to the nearest point in the other set [10]. More formally, Hausdorff distance between two edge points sets O_i and O_j is the maximum function defined in Eq. 1.

$$H(O_i, O_j) = max\{min\{d(a,b)\}\} \tag{1}$$

Where $a \in O_i$ and $b \in O_j$ Thus, the operational agent assigns to the matrix elements different values according to Eq. 2 as follows:

$$M(i,j) = 1 \quad if\, H(O_i, O_j) \leq \omega \quad otherwise \quad 0 \tag{2}$$

The parameter ω is a fixed threshold. Once M constructed, we can then pass to the next step of clustering. In, the second step we look at the ”optimal” number of clusters needed for the next concepts learning. According to a column k, the existence at least one element with value 1 at a row l yields the construction of a new cluster j.

For this same column k, if there exits more elements with values 1, then all other objects representing the rows are enclosed in the same cluster j (Eq. 3). In case, all the elements of a column are set to zero this implies that the object of this column is already contained in a previous cluster.

$$if \quad M(k,l) = 1 \quad then \quad \{O_k \in S_j \wedge O_l \in S_j\} \tag{3}$$

When all matrix columns were visited, then the optimal number of clusters is determined. The following algorithm outlines the formation of clusters.

```
   Clustering Algorithm
// In [$M$: matrix $(nxn)$]
Begin
$j = 0$;   $k = 1$;   // $k$ is the column index
While ($k \leq n$) do
If (($k > 1) and\, O_k \in \{S_r/r = 1 \dots j-1\}\}$   $then$   $k = k+1$;
Else
$j = j+1$;   $S_j = \emptyset$;
For each row $l = 1$ to $n$ do
If $M(k,l) = 1$ then $S_j = S_j \cup O_l$;
// Out [$S$: set]
End.
```

3.2 Evolutionary Agent

Representing concepts as sets of rules has long been popular in machine learning, because, among other properties, rules are easy to represent and humans can interpret them easily. In evolutionary algorithms there are two main ways to represent rule sets: the "Michigan" approach exemplified by Holland classifier [6], [7], [9] and the "Pittsburgh" approach exemplified by Smith LS-1 system [5], [8], [11], [20]. In our studies, we adopt the Michigan approach to encode a rule. Thus, an individual in its general form is a set of items of $attribute_i \in [l_i, u_i]$,where $attribute_i$is the i^{th}numeric attribute in the rule template from left to right. The association rules shown to the environment have the form:

$\{(AT1_1,AT2_1),(AT1_2,AT2_2),\ldots,(AT1_k,AT2_k)\} \Rightarrow \{AT1_m\}$. As instance, we can consider a simple context of image including primitive figures like geometric forms. Thus, if we consider the following:

Domain :$AT1_i \in \{triangle, circle, square, rectangle, hexagon, ellipse\}$
$AT2_i \in [1,10]$

Example of coding rule. $\underbrace{0 \quad 0 \quad 1 \quad 0 \quad 0}_{\text{rectangle}} \quad 4 \quad \underbrace{9}_{\text{chair}}$

The evolutionary algorithm we are using is described below.

```
Algorithm EvoAlg
// in [t:file of encoded-examples]
Begin
   Initializepopulation(p)
   For $i = 1$ to max-generations do
      Evaluate(p);
      next.p:=Selectthebestof(p);
      next.p:=next.p+Replicate(p);
      next.p:=next.p+Recombine(p);
      p:=next.p;
   End
End // Selectthebestof(p); out [r:rules]
```

Equation 4 gives the fitness function $f(\tau)$used during the evaluation process. The greater the value, the better the individual is.

$$f(\tau) = N - CE(\tau) + G(\tau) + coverage(\tau) \tag{4}$$

In Eq. 4,τ is an individual;N is the number of examples; CE is the concept error, i.e. the number of examples belonging to the region defined by the ruleτ, that do not have the same concept (class); $G(\tau)$,is the number of examples correctly "classified", and $coverage(\tau)$ gives the size proportion correctly "classified" [1].

4 Conclusion

In this paper, we presented a cognitive architecture inspired from human characteristics to deal with the visual cognitive modality. The system includes many reasoning paradigms such as image processing, evolutionary computing and image mining. Our efforts were based on the conceptualization of the cognitive framework for detecting and understanding objects in several contexts. Obviously, the system relies on knowledge discovering using association rules from databases. Results obtained so far look promising but we need to improve several aspects in our research efforts.

References

1. Aguilar-Ruiz, J.S., Riquelme, J.C., Toro, M.: Evolutionary learning of hierarchical decision rules. IEEE Transactions on Man and Cybernetics, Part B 33(2), 324–331 (2003)
2. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between set of items in large databases. In: Proc. ACM SIGMOD Int. Conf. on Management of data, pp. 207–216 (1993)
3. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large database. In: Proceedings of 20th International Conference on Very Large Data Bases, Santiago Chile (1994)
4. Bhanu, B., Lin, Y., Krawiek, K.: Evolutionary synthesis of pattern recognition systems. Springer, Heidelberg (2005)
5. De Jong, K.A., Spears, W.M., Gordon, D.F.: Using genetic algorithms for concept learning. Machine Learning, 161–188 (1993)
6. Giordana, A., Neri, F.: Search intensive concept induction. Evolutionary Computation 3(4), 376–416 (1996)
7. Greene, D.P., Smith, S.F.: Competition-based induction of decision models from examples. Machine Learning 13, 229–257 (1993)
8. Hekanaho, J.: GA-based rule enhancement concept learning. In: Proc. Third International Conference on Knowledge Discovery and Data Mining, pp. 183–186 (1997)
9. Holland, J.: Escaping brittleness. The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In: Machine Learning: An Artificial Intelligence Approach (1986)
10. Huttenlocher, D.P., Rucklidge, W.J.: A multi-resolution technique for comparing images using Hausdorff distance. Dpt. of CS, Cornell University (1994)
11. Janikow, C.Z.: A knowledge intensive genetic algorithm for supervised learning. Machine Learning 13, 189–228 (1993)
12. Krawiec, K.: Learning high-level visual concepts using attributed primitives and genetics programming. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 515–519. Springer, Heidelberg (2006)
13. Krawiec, K.: Evolutionary learning of primitive-based visual concepts. In: Proc. IEEE Congress on Evolutionary Computation, Vancouver, Canada, pp. 1308–1315 (2006)
14. Krawiec, K., Bhanu, B.: Visual learning by coevolutionary features synthesis. IEEE Transactions on Systems Man and Cybernetics Part B 35, 409–425 (2005)

15. Maloof, M., Langley, P., Binford, P., Nevatia, R., Sage, S.: Improved rooftop detection in aerial images with machine learning. Machine Learning 53, 157–1991 (2003)
16. Ogiela, M., Tadeusiewic, Z.R.: Nonlinear processing and semantic content analysis in medical imaging - a cognitive approach. IEEE Transactions on Instrumentation and Measurements 54, 2149–2155 (2005)
17. Rizki, M., Zmuda, M., Tamburino, L.: Evolving pattern recognition systems. IEEE transactions on Evolutionary Computation 6, 594–609 (2002)
18. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules. In: Proceedings of the VLDB Conference, Switzerland, pp. 432–444 (1995)
19. Segen, J.: GEST: a learning computer vision system that recognizes hand gestures. Machine Learning: A Multistrategy Approach 4, 621–634 (1994)
20. Smith, S.: Flexible learning of problem solving heuristics through adaptive search. In: Proc. Eight International Joint Conference on Artificial Intelligence, pp. 422–425 (1983)
21. Teller, A., Veloso, M.: PADO: a new learning architecture for object recognition. In: Symbolic Visual Learning, pp. 77–112. Eds. Oxford Press, New York (1997)

Author Index