Studies in Computational Intelligence 446

Giancarlo Fortino
Costin Bădică
Michele Malgeri
Rainer Unland (Eds.)

Intelligent Distributed Computing VI

Proceedings of the 6th International Symposium on Intelligent Distributed Computing - IDC 2012 Calabria, Italy, September 2012

Springer

Studies in Computational Intelligence 446

For further volumes:
http://www.springer.com/series/7092

Giancarlo Fortino, Costin Bădică, Michele Malgeri,
and Rainer Unland (Eds.)

Intelligent Distributed Computing VI

Proceedings of the 6th International Symposium on Intelligent Distributed Computing – IDC 2012, Calabria, Italy, September 2012

Springer

Editors
Giancarlo Fortino
Department of Electronics, Informatics and Systems
University of Calabria
Rende
Italy

Michele Malgeri
Dipartimento di Ingegneria Elettrica, Elettronica e Informatica
Università di Catania
Catania
Italy

Costin Bădică
Software Engineering Department
Faculty of Automatics, Computers and Electronics
University of Craiova
Craiova
Romania

Rainer Unland
Institute for Computer Science and Business Information Systems (ICB)
University of Duisburg-Essen
Essen
Germany

ISSN 1860-949X
e-ISSN 1860-9503
ISBN 978-3-642-32523-6
e-ISBN 978-3-642-32524-3
DOI 10.1007/978-3-642-32524-3
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012944263

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The emergent field of Intelligent Distributed Computing focuses on the development of a new generation of intelligent distributed systems. It faces the challenges of adapting and combining research in the fields of Intelligent Computing and Distributed Computing. Intelligent Computing develops methods and technology ranging from classical artificial intelligence, computational intelligence and multi-agent systems to game theory. The field of Distributed Computing develops methods and technology to build systems that are composed of collaborating distributed components. 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 2012 continues the tradition of the IDC Symposium Series that 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 Intelligent Computing, with scientific merit in at least one of these two areas. IDC 2012 was the sixth event in the series and was organized by *University of Calabria*, *University of Craiova*, and *SenSysCal S.r.l.* in *Calabria, Italy* during September 24-26, 2012. IDC 2012 had a special interest in novel architectures and methods which facilitate intelligent distributed solutions to complex problems by combining human cognitive capabilities and automated processing.

IDC 2012 was collocated with: (i) A4C (Agents for Cloud) 2012 workshop; (ii) 4^{th} International Workshop on Multi-Agent Systems Technology and Semantics, MASTS 2012.

The material published in this book is divided into three main parts: (i) 30 contributions of IDC 2012 participants; (ii) 4 contributions of A4C 2012 participants; (iii) 3 contributions of MASTS 2012 participants.

The response to IDC 2012 call for paper was generous. We received 45 submissions from 16 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 2012 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 12 regular papers and 17 short papers were selected for presentation and were included in this volume, resulting in acceptance rates of 26.7 % for regular papers and 37.8 % for short papers. Additionally, A4C 2012 and MASTS 2012 workshops received 8 submissions each. After the careful review (each submission was reviewed by at least 3 members of the A4C 2012 and MASTS 2012 Program Committee), 4 papers for A4C 2012 and 3 papers for MASTS 2012 were selected for presentation and were included in this book.

The 37 contributions published in this book address many topics related to theory and applications of intelligent distributed computing and multi-agent systems, including: self-organization, bio-inspiration, adaptive and autonomous distributed systems, agent programming, parallel computing, social computing and networks, agents and sensor networks, collaborative computing, mobile networks and intelligence, dynamic reasoning, agent-based intelligent applications, cloud computing, and smart environments.

We would like to thank to Janusz Kacprzyk, editor of *Studies in Computational Intelligence* series and member of the Steering Committee for his continuous support and encouragement for the development of the IDC Symposium Series. We would like to thank to the IDC 2012, A4C 2012 and MASTS 2012 Program Committee members for their work in promoting the event and refereeing submissions and also to all colleagues who submitted papers to IDC 2012, A4C 2012 and MASTS 2012. We deeply appreciate the efforts of our invited speakers Belur V. Dasarathy and Andrea Omicini and thank them for their interesting lectures and Giovanni Caire for giving an interesting tutorial on WADE. Special thanks also go to organizers of MASTS 2012: Adina Magda Florea, John Jules Meyer, and Amal El Fallah Seghrouchni, and of A4C: Salvatore Venticinque. Finally, we appreciate the efforts of local organizers on behalf of *SenSysCal S.r.l.* and *University of Calabria* in *Calabria, Italy* for organizing IDC 2012, A4C 2012 and MASTS 2012.

Rende (CS), Italy
July 2012

Giancarlo Fortino
Costin Bădică
Michele Malgeri
Rainer Unland

Organization

Organizers

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica (DIMES), University of Calabria, Italy
Software Engineering Department, Faculty of Automation, Computers and Electronics, University of Craiova, Romania
SenSysCal S.r.l., Italy

Conference Chair

Giancarlo Fortino, University of Calabria, Italy

Steering Committee

Janusz Kacprzyk, Polish Academy of Sciences, Poland
Costin Bădică, University of Craiova, Romania
Michele Malgeri, University of Catania, Italia
George A. Papadopoulos, University of Cyprus, Cyprus
Marcin Paprzycki, Polish Academy of Sciences, Poland
Frances Brazier, Delft University of Technology, The Netherlands
Kees Nieuwenhuis, D-CIS Lab / Thales Research & Technology, The Netherlands
Giancarlo Fortino, University of Calabria, Italy
Mohammad Essaaidi, Abdelmalek Essaadi University in Tetuan, Morocco

Invited Speakers

Belur V. Dasarathy, Consultant, Information Fusion Technologies
Andrea Omicini, University of Bologna, Italy

Program Committee Chairs

Giancarlo Fortino, University of Calabria, Italy
Michele Malgeri, University of Catania, Italy
Rainer Unland, University of Duisburg-Essen, Germany

Program Committee

Salvador Abreu, Universidade de Évora and CENTRIA FCT/UNL, Portugal
Răzvan Andonie, Central Washigton University, USA and University of Transylvania, Romania
Costin Bădică, University of Craiova, Romania
Mert Bal, Miami University, USA
Amar Balla, LCMS-ESI Ecole nationale Supérieure d'Informatique, Algeria
Nick Bassiliades, Aristotle University of Thessaloniki, Greece
Doina Bein, ARL, Pennsylvania State University, USA
Nik Bessis, University of Derby, UK
Lars Braubach, University of Hamburg, Germany
Dumitru Dan Burdescu, University of Craiova, Romania
Giacomo Cabri, University of Modena and Reggio Emilia, Italy
David Camacho, Universidad Autonoma de Madrid, Spain
Mario Cannataro, University Magna Graecia of Catanzaro, Italy
Vincenza Carchiolo, University of Catania, Italy
Jen-Yao Chung, IBM T.J. Watson Research Center, USA
Phan Cong-Vinh, Centre for Applied Formal Methods, London South Bank University, UK
Massimo Cossentino, ICAR-CNR, Italy
Alfredo Cuzzocrea, University of Calabria, Italy
Paul Davidsson, Blekinge Institute of Technology, Sweden
Beniamino Di Martino, Second University of Naples, Italy
Giuseppe Difatta, University of Reading, UK
Luminiţa Dumitriu, "Dunarea de Jos" University of Galaţi, Romania
Amal El Fallah Seghrouchni, LIP6 - University of Pierre and Marie Curie, France
George Eleftherakis, CITY International Faculty, University of Sheffield, Greece
Vadim A. Ermolayev, Zaporozhye National University, Ukraine
Mohammad Essaaidi, Abdelmalek Essaadi University in Tetuan, Morocco
Mostafa Ezziyyani, Université Abdelmalek Essaadi, Maroc
Adina Magda Florea, "Politehnica" University of Bucharest, Romania
Giancarlo Fortino, University of Calabria, Italy
Stefano Galzarano, University of Calabria, Italy
Maria Ganzha, Elblag University of Humanities and Economics, Poland
Alfredo Garro, University of Calabria, Italy
Raffaele Gravina, University of Calabria, Italy
Nathan Griffiths, University of Warwick, UK
Antonio Guerrieri, University of Calabria, Italy
Michael Hartung, University of Leipzig, Germany
Tom Holvoet, K.U.Leuven, Belgium
Barna Laszlo Iantovics, Petru Maior University of Targu Mures, Romania
Mirjana Ivanović, University of Novi Sad, Serbia
Jason J. Jung, Yeungnam University, South Korea

Igor Kotenko, Russian Academy of Sciences, Russia
Dariusz Krol, Wroclaw University of Technology, Poland
Florin Leon, Technical University "Gheorghe Asachi" of Iasi, Romania
Wen-Feng Li, Wuhan University of Technology, China
Alessandro Longheu, University of Catania, Italy
Heitor Silverio Lopes, Federal University of Technology - Parana, Brazil
José Machado, University of Minho, Portugal
Michele Malgeri, University of Catania, Italy
Giuseppe Mangioni, University of Catania, Italy
Yannis Manolopoulos, Aristotle University of Thessaloniki, Greece
Gleizes Marie-Pierre, Institut de Recherche en Informatique de Toulouse, France
Viviana Mascardi, University of Genova, Italy
Amnon Meisels, Ben-Gurion University, Israel
Grzegorz J. Nalepa, AGH University of Science and Technology, Kraków, Poland
Viorel Negru, Western University of Timişoara, Romania
Peter Noerr, MuseGlobal, Inc., USA
Eugenio Oliveira, Universidade do Porto, Portugal
Andrea Omicini, University of Bologna, Italy
Mihaela Oprea, University Petroleum-Gas of Ploieşti, Romania
Marcin Paprzycki, Polish Academy of Sciences, Poland
Gregor Pavlin, D-CIS Lab / Thales Research & Technology, Netherlands
Juan Pavon, Universidad Complutense Madrid, Spain
Stefan-Gheorghe Pentiuc, University "Stefan cel Mare" Suceava, Romania
Dana Petcu, Western University of Timişoara, Romania
Florin Pop, University "Politehnica" of Bucharest, Romania
Radu-Emil Precup, "Politehnica" University of Timişoara, Romania
Shahram Rahimi, Southern Illinois University, USA
Domenico Rosaci, University of Reggio Calabria, Italy
Wilma Russo, University of Calabria, Italy
Ioan Salomie, Technical University of Cluj-Napoca, Romania
Corrado Santoro, University of Catania, Italy
Murat Sensoy, University of Aberdeen, UK
Weiming Shen, NRC, Canada
Safeeullah Soomro, Yanbu University College, Saudi Arabia
Giandomenico Spezzano, University of Calabria, Italy
Rainer Unland, University of Duisburg-Essen, Germany
Salvatore Venticinque, Second University of Naples, Italy
Laurent Vercouter, Ecole des Mines de St-Etienne, France
Lucian Vinţan, Academy of Technical Sciences, Romania
Martijn Warnier, Delft University of Technology, The Netherlands
Niek Wijngaards, D-CIS Lab / Thales Research & Technology, Netherlands
Filip Zavoral, Charles University, Czech Republic

Organizing Committee

Giancarlo Fortino, University of Calabria, Italy
Costin Bădică, University of Craiova, Romania
Stefano Galzarano, University of Calabria, Italy
Raffaele Gravina, University of Calabria, Italy
Antonio Guerrieri, University of Calabria, Italy

A4C 2012 Workshop Chairs

Salvatore Venticinque, Second University of Naples, Italy

A4C 2012 Workshop Program Committee

Rocco Aversa, Second University od Naples, Italy
Dana Petcu, Institute e-Austria Timisoara (IEAT), Romania
Antonino Mazzeo, University of Naples Federico II, Italy
Nicola Mazzocca, University of Naples Federico II, Italy
Marcin Paprzycki, Systems Research Institute of the Polish Academy of Sciences, Poland
Marios Dikaiakos, University of Cyprus, Cyprus
Omer F. Rana, Cardiff University, United Kingdom
Leonard Barolli , Fukuoka Institute of Technology, Japan
Fatos Xhafa, Universitat Politcnica de Catalunya, Spain

MASTS 2012 Workshop Chairs

Adina Magda Florea, University "Politehnica" of Bucharest, Romania
Amal El Fallah Seghrouchni, Université Pierre & Marie Curie, France
John Jules Meyer, Universiteit Utrecht, The Netherlands

MASTS 2012 Workshop Program Committee

Costin Bădică, University of Craiova, Romania
Olivier Boissier, ENS des Mines Saint-Etienne, France
Adina Magda Florea, University Politehnica of Bucharest, Romania
Pascale Kuntz-Cosperec, Polytech'Nantes, France
John Jules Meyer, Utrecht University, The Netherlands
Andrei-Horia Mogos, University Politehnica of Bucharest, Romania
Viorel Negru, West University of Timisoara, Romania
Sascha Ossowski, University Rey Juan Carlos, Spain
Marcin Paprzycki, Polish Academy of Science, Poland
*Amal El Fallah Seghrouchn*i, Université Pierre & Marie Curie, France
Stefan Trausan-Matu, University Politehnica of Bucharest, Romania
Laurent Vercouter, INSA Rouen, France
Antoine Zimmermann, ENS des Mines Saint-Etienne, France

Contents

Nature-Inspired Coordination for Complex Distributed Systems

Andrea Omicini

Abstract. Originating from closed parallel systems, coordination models and technologies gained in expressive power so to deal with complex distributed systems. In particular, nature-inspired models of coordination emerged in the last decade as the most effective approaches to tackle the complexity of pervasive, intelligent, and self-* systems. In this paper we survey the most relevant nature-inspired coordination models, discuss the main open issues, and explore the trends for their future development.

1 Introduction

Coordination models essentially address the issue of how to harness the complexity of the interaction space in the design of distributed systems [26]. Even though originating in the context of closed, parallel systems [6], coordination models soon gained relevance and power in those scenarios where complexity is a key factor—such as intelligent, knowledge-intensive, pervasive, self-organising systems [3].

Coordination, indeed, is not a matter of computational systems only. In fact, it concerns complex systems of any sort: roughly speaking, it is how complex system get together so as to work [17]. Many natural systems are prominent examples of complex systems exhibiting models and patterns of coordination that enforce desirable properties for computational systems such as fault-tolerance, adaptiveness, and self-organisation. Accordingly, a whole class of coordination models is inspired by the extraction of abstractions, patterns, and mechanisms out of natural systems: *nature-inspired* coordination models come from the idea that working complex systems exist in the real world, which we can observe so as to understand their basic principles and mechanisms, to abstract them, and to bring them within our computational systems [28]. In particular, understanding the principles and

Andrea Omicini
DISI, ALMA MATER STUDIORUM – Università di Bologna, Italy
e-mail: andrea.omicini@unibo.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 1–6.
springerlink.com

mechanisms of coordination within complex natural systems is seemingly an effective approach for the definition of coordination models and computational technologies for complex distributed systems.

2 Nature-Inspired Coordination: Early Examples

Historically, nature-inspired models of coordination are grounded in studies on the behaviour of social insects, like ants or termites. In [8], Grassé noted that in termite societies *"The coordination of tasks and the regulation of constructions are not directly dependent from the workers, but from constructions themselves."*, and introduced the notion of *stigmergy* as the fundamental coordination mechanism. For instance, in ant colonies, pheromones act as environment markers for specific social activities, and drive both the individual and the social behaviour of ants.

Nowadays, stigmergy generally refers to a set of nature-inspired coordination mechanisms mediated by the *environment*: digital pheromones [19] and other *signs* made and sensed in a shared environment [18] can be exploited for the engineering of adaptive and self-organising computational systems. By interpreting tuples as signs, and tuple spaces as environment abstractions, tuple-based coordination models [21] – derived from the original LINDA model [6] – can be used to implement stigmergic coordination within complex distributed systems [9, 20].

More or less contemporary with LINDA is the *chemistry-inspired coordination* model Gamma [1]: as for the CHAM (chemical abstract machine) model [2], coordination in Gamma is conceived as the evolution of a space governed by chemical-like rules, globally working as a rewriting system.

On the other hand, *field-based coordination* models like TOTA [9] are inspired by the way masses and particles move and self-organise according to gravitational/electromagnetic fields [10]. There, computational force fields in the form of distributed tuples, generated either by the active components or by the pervasive coordination infrastructure, propagate across the environment, and drive the actions and motion of the component themselves.

Some of the basic issues of complex system coordination straightforwardly emerge from the above-mentioned models. First, the role of the *environment* is essential in nature-inspired coordination. Components of a distributed system are not limited to direct interaction, but they can communicate and coordinate *indirectly* through the environment. Also, the environment is *active*: it exhibits an autonomous dynamics (as in the case of chemical-like mechanisms) and affects component coordination (as in the case of pheromone-like mechanisms). Finally, the environment has a *structure* (as in the case of field-based models): interaction in complex distributed systems can then be enriched with some notion of *locality*, and components of any sort can *move* through a *topology*.

3 Nature-Inspired Coordination: Issues

Capturing *some* of the principles and mechanisms of natural systems does not ensure to capture their essence. For instance, chemical coordination models such as Gamma and CHAM exploit the raw schema of computation as chemical reaction, but are not expressive enough to fully reproduce any non-trivial chemical system. In fact, even the simplest model for real chemical reactions requires a notion of *reaction rate* (roughly speaking, a measure of how fast a reaction takes place): since neither Gamma nor CHAM provide for such a notion, they are not endowed with the expressive power required to build computational systems fully matching the behaviour of real chemical systems.

Studies on the notion of *self-organising coordination* [24] point out some general issues of nature-inspired coordination of complex systems—in particular when compared with "classical" coordination models for distributed systems. In fact, most of the traditional coordination models feature abstractions enacting coordination laws that are typically reactive, (essentially) deterministic, and global as well. In complex systems featuring self-* properties, instead, coordination patterns typically appear at the global level by emergence, from probabilistic, time-dependent coordination laws based on local criteria.

In particular, many coordination models either implicitly or explicitly recognise that full expressiveness requires addressing the issues of *time dependency* and *stochasticity*. Among the first attempts, StoKlaim [4] – a stochastic extension of the Linda-derived Klaim model for mobile coordination [3] – adds distribution rates to coordination primitives, thus making it possible the modelling of non-deterministic real-life phenomena such as failure rates and inter-arrival times.

In its turn, SwarmLinda aims at enhancing Linda implementation with swarm intelligence [22] in order to achieve many natural-system features such as scalability, adaptiveness, and fault-tolerance. This is obtained by modelling tuple templates as ants, and enhancing them with a probabilistic behaviour when looking for matching tuples in a distributed setting.

Time dependency is generally addressed by the time-aware extension of ReSpecT [14]. There, the ReSpecT language for programming logic tuple centres [2] is extended in order to catch with time, and to support the definition and enforcement of *timed coordination policies*. ReSpecT programmed tuple centres can then work as time-dependent abstractions for the coordination of distributed processes, once deployed upon the TuCSoN coordination infrastructure [15].

In the overall, however, the above-mentioned models for the coordination of complex distributed system fail to provide a comprehensive and exhaustive approach that could capture all the essential features of nature-inspired coordination. Instead, this is precisely the main target of many novel research lines, which stretch existing coordination modes (typically, tuple-based ones) to make them reach their full potential, trying to achieve the expressive power required to model and build distributed systems with a complexity comparable to natural systems.

4 Nature-Inspired Coordination: Trends

The most relevant trend in today models for the coordination of complex distributed system is aimed at expressing the *full dynamics* of complex natural systems. Along this line, *chemical tuple spaces* [23] represent a successful attempt to exploit the chemical metaphor at its full extent. There, data, devices, and software agents are represented in terms of tuples, and system behaviour is expressed by means of chemical-like laws – enacting interaction patterns such as composition, aggregation, competition, contextualisation, diffusion, and decay – which are actually time-dependent and stochastic. Chemical laws, in fact, are implemented based on the Gillespie's algorithm [1] using *uniform coordination primitives* – that is, LINDA-like coordination primitives returning tuples matching a template with a uniform distribution [5] – in order to capture the full-fledged dynamics of real chemical systems within the coordination abstractions.

Blending metaphors from different sources represents another fundamental trend. For instance, the SAPERE coordination model for pervasive service ecosystems [27] exploits the chemical metaphor for driving the evolution of coordination abstractions, biochemical abstractions for topology and diffusion, and the notion of ecosystem altogether in order to model the overall system structure and dynamics. There, all the components (data, devices, agents) are uniformly represented through special tuples called LSA (Live Semantic Annotations) which can chemically bond to each other, live within sorts of biochemical compartments (LSA-spaces), and evolve based on natural laws of the ecosystem, called eco-laws [25].

Finally, integrating nature-inspired with knowledge-oriented coordination is the last fundamental trend. Both chemical tuple spaces and SAPERE abstractions relay on the semantic interpretation of coordination items—in the same way as *semantic tuple centres* [12]. On the other hand, a nature-inspired coordination model focussing on knowledge management is MoK (Molecules of Knowledge) [11]. There, knowledge sources produce atoms of knowledge in biochemical compartments, which then aggregate in molecules and diffuse according to biochemical reactions. In MoK, the full power of the biochemical metaphor is exploited in order to achieve knowledge self-organisation within knowledge-intensive environments.

5 Conclusion

Starting from early chemical and stigmergic approaches, nature-inspired models of coordination deeply evolved aimed at becoming the core of the engineering of complex distributed systems, such as pervasive, knowledge-intensive, intelligent, and self-* systems. In this paper we shorty survey their early history, devise their main issues, and point out the most promising trends—namely, full metaphor adoption, heterogeneous metaphor blending, and knowledge-oriented integration.

In the overall, nature-inspired models of coordination already have a long history behind them, and a huge potential for development still to be explored.

Acknowledgements. This work has been supported by the EU-FP7-FET Proactive project SAPERE Self-aware Pervasive Service Ecosystems, under contract no. 256873.

References

1. Banâtre, J.-P., Fradet, P., Le Métayer, D.: Gamma and the Chemical Reaction Model: Fifteen Years After. In: Calude, C.S., Păun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 17–44. Springer, Heidelberg (2001), doi:10.1007/3-540-45523-X_2
2. Berry, G.: The chemical abstract machine. Theoretical Computer Science 96(1), 217–248 (1992), doi:10.1016/0304-3975(92)90185-I
3. De Nicola, R., Ferrari, G., Pugliese, R.: KLAIM: A kernel language for agent interaction and mobility. IEEE Transaction on Software Engineering 24(5), 315–330 (1998), doi:10.1109/32.685256
4. De Nicola, R., Latella, D., Katoen, J.P., Massink, M.: StoKlaim: A stochastic extension of Klaim. Tech. Rep. 2006-TR-01, Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo", ISTI (2006)
5. Gardelli, L., Viroli, M., Casadei, M., Omicini, A.: Designing Self-organising MAS Environments: The Collective Sort Case. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 254–271. Springer, Heidelberg (2007)
6. Gelernter, D.: Generative communication in Linda. ACM Transactions on Programming Languages and Systems 7(1), 80–112 (1985), doi:10.1145/2363.2433
7. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81(25), 2340–2361 (1977), doi:10.1021/j100540a008
8. Grassé, P.P.: La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Insectes Sociaux 6(1), 41–80 (1959), doi:10.1007/BF02223791
9. Mamei, M., Zambonelli, F.: Programming pervasive and mobile computing applications with the TOTA middleware. In: Proceedings of 2nd IEEE Annual Conference (PerCom 2004), Pervasive Computing and Communications, Orlando, FL, USA, March 14-17, pp. 263–273 (2004), doi:10.1109/PERCOM.2004.1276864
10. Mamei, M., Zambonelli, F.: Field-Based Coordination for Pervasive Multiagent Systems. In: Models, Technologies, and Applications. Springer Series in Agent Technology. Springer (2006), doi:10.1007/3-540-27969-5
11. Mariani, S., Omicini, A.: Molecules of Knowledge: Self-organisation in Knowledge-Intensive Environments. In: Fortino, G., et al. (eds.) Intelligent Distributed Computing VI. SCI, vol. 446, pp. 17–22. Springer, Heidelberg (2012)
12. Nardini, E., Omicini, A., Viroli, M., Schumacher, M.I.: Coordinating e-health systems with TuCSoN semantic tuple centres. Applied Computing Review 11(2), 43–52 (2011), doi:10.1145/1964144.1964150
13. Omicini, A., Denti, E.: From tuple spaces to tuple centres. Science of Computer Programming 41(3), 277–294 (2001), doi:10.1016/S0167-6423(01)00011-9
14. Omicini, A., Ricci, A., Viroli, M.: Time-Aware Coordination in ReSpecT. In: Jacquet, J.-M., Picco, G.P. (eds.) COORDINATION 2005. LNCS, vol. 3454, pp. 268–282. Springer, Heidelberg (2005)

15. Omicini, A., Ricci, A., Viroli, M.: Timed environment for Web agents. Web Intelligence and Agent Systems 5(2), 161–175 (2007), http://iospress.metapress.com/content/bv82m5w0056m3877/
16. Omicini, A., Viroli, M.: Coordination models and languages: From parallel computing to self-organisation. The Knowledge Engineering Review 26(1), 53–59 (2011), doi:10.1017/S026988891000041X
17. Ossowski, S., Menezes, R.: On coordination and its significance to distributed and multi-agent systems. Concurrency and Computation: Practice and Experience 18(4), 359–370 (2006), doi:10.1002/cpe.943, Special Issue: Coordination Models and Systems
18. Van Dyke Parunak, H.: A Survey of Environments and Mechanisms for Human-Human Stigmergy. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 163–186. Springer, Heidelberg (2006), doi:10.1007/11678809_10
19. Van Dyke Parunak, H., Brueckner, S., Sauter, J.: Digital pheromone mechanisms for coordination of unmanned vehicles. In: Castelfranchi, C., Johnson, W.L. (eds.) 1st International Joint Conference on Autonomous Agents and Multiagent Systems, vol. 1, pp. 449–450. ACM, New York (2002), doi:10.1145/544741.544843
20. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive Stigmergy: Towards a Framework Based on Agents and Artifacts. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 124–140. Springer, Heidelberg (2007)
21. Rossi, D., Cabri, G., Denti, E.: Tuple-based technologies for coordination. In: Omicini, A., Zambonelli, F., Klusch, M., Tolksdorf, R. (eds.) Coordination of Internet Agents: Models, Technologies, and Applications, ch. 4, pp. 83–109. Springer (2001)
22. Tolksdorf, R., Menezes, R.: Using Swarm Intelligence in Linda Systems. In: Omicini, A., Petta, P., Pitt, J. (eds.) ESAW 2003. LNCS (LNAI), vol. 3071, pp. 49–65. Springer, Heidelberg (2004)
23. Viroli, M., Casadei, M., Nardini, E., Omicini, A.: Towards a Pervasive Infrastructure for Chemical-Inspired Self-organising Services. In: Weyns, D., Malek, S., de Lemos, R., Andersson, J. (eds.) SOAR 2009. LNCS, vol. 6090, pp. 152–176. Springer, Heidelberg (2010)
24. Viroli, M., Casadei, M., Omicini, A.: A framework for modelling and implementing self-organising coordination. In: Shin, S.Y., Ossowski, S., Menezes, R., Viroli, M. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), vol. III, pp. 1353–1360. ACM, Honolulu (2009), doi:10.1145/1529282.1529585
25. Viroli, M., Pianini, D., Montagna, S., Stevenson, G.: Pervasive ecosystems: a coordination model based on semantic chemistry. In: Ossowski, S., Lecca, P., Hung, C.C., Hong, J. (eds.) 27th Annual ACM Symposium on Applied Computing (SAC 2012). ACM, Riva del Garda (2012)
26. Wegner, P.: Why interaction is more powerful than algorithms. Communications of the ACM 40(5), 80–91 (1997), doi:10.1145/253769.253801
27. Zambonelli, F., Castelli, G., Ferrari, L., Mamei, M., Rosi, A., Di Marzo, G., Risoldi, M., Tchao, A.E., Dobson, S., Stevenson, G., Ye, Y., Nardini, E., Omicini, A., Montagna, S., Viroli, M., Ferscha, A., Maschek, S., Wally, B.: Self-aware pervasive service ecosystems. Procedia Computer Science 7, 197–199 (2011), doi:10.1016/j.procs.2011.09.006, Proceedings of the 2nd European Future Technologies Conference and Exhibition 2011 (FET 2011)
28. Zambonelli, F., Viroli, M.: A survey on nature-inspired metaphors for pervasive service ecosystems. International Journal of Pervasive Computing and Communications 7(3), 186–204 (2011), doi:10.1108/17427371111172997

Information Fusion as Aid for Qualitative Enhancement of Intelligence Extraction in Multi-source Environments - A Panoramic View of Architectures, Algorithms and Applications

Belur V. Dasarathy

Abstract. This invited talk highlights the positive role of information fusion in distributed sensor networks. The presentation will begin with a brief introduction to the twin topics namely, sensor networks and information fusion. This is followed by an overview of how the objectives of the former can be aided by the advancements in the latter domain. This will include a brief overview of sensor network characteristics, its taxonomy, and performance metrics followed by a parallel overview of the field of information fusion. This will lead into the delineation of the role of information fusion in meeting specific objectives of sensor networks. Next, a global, flexible Information fusion architecture model that maximizes the overall opportunity for synergistic exploitation of sensor network potential in terms of the objectives that are aided by the fusion process will be presented and discussed in greater detail. This architecture conceives the fusion function as an input-output process that can be embedded at various points (nodes) in the hardware/software system underpinning the sensor networks.

Belur V. Dasarathy
Consultant, Information Fusion and Decision System Technologies
Editor-in-Chief, Information Fusion
e-mail: fusion-consultant@ieee.org, belur.d@gmail.com
http://belur.no-ip.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, p. 7.
springerlink.com

Using Building Blocks for Pattern-Based Simulation of Self-organising Systems

Christopher Haubeck, Ante Vilenica, and Winfried Lamersdorf

Abstract. The constantly rising complexity of distributed systems and an increasing demand for non-functional requirements lead to approaches featuring *self-organising* characteristics. Developing these systems is challenged by their hardly predictable dynamics and emergent phenomena and requires therefore the incorporation of *simulation* techniques. In doing so, not all needed development activities can be realised by just one software application because self-organisation often implies unique settings, goals, and development methods as well as the use of individual code sections. In order to handle such unique environments, this contribution presents a pattern-based concept that incorporates reusable patterns for different development issues of self-organising systems by encapsulating various methods, algorithms, and applications in so called *building blocks* and combining them in a coherent and hierarchical process.

1 Introduction

Old-fashioned concepts of developing distributed computer systems top-down with centralised control are not suitable for many new application domains which are characterised by high complexity, diversity, and heterogeneity of their components. Such applications as, e.g., in mobile / pervasive computing, typically exhibit a dynamic behaviour that is hardly predictable and requires fast adaption to changing application contexts that may even be unknown a priori. For such applications *Self-Organisation* (SO) has proven to be a promising approach that can deal with these challenges autonomously by (self-)adapting their respective structure and behaviour without any centralised or external control. However, there are three facts that challenge the purposeful development of self-organising systems: (I) the inherent bottom-up development process, (II) local interactions among components

Christopher Haubeck · Ante Vilenica · Winfried Lamersdorf
Distributed Systems, Informatics Department, University of Hamburg
e-mail: {haubeck,vilenica,lamersdorf}@informatik.uni-hamburg.de

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 9–15.
springerlink.com

that cause effects on global level and (III) the presence of emergent phenomena. To overcome these challenges [2] propose a simulation-based development process for SO, i.e. simulation becomes an inherent step during an iterative development as it evaluates whether the designed components produce the designated behaviour.

In terms of the considered system each simulation can be very distinct, e.g., settings, goals, used simulation tools and languages. For this reason the process of performing simulation studies can be considered as a unique task, which in most cases cannot be handled by just one software system. As a result the development of self-organising systems takes place in a heterogeneous and potentially distributed environment of various systems. At the same time the task of *coordinating* different software frameworks to achieve the designated goal is quite demanding and requires a lot of *manual* effort by the developer [6]. In this regard this novel concept of *building blocks* offers the possibility of developing and using patterns that guide the development process of self-organising systems. Thereby, this approach combines the flexibility of individual development processes with the exchangeability and reusability of standardised processes and services.

The remainder of this paper is organised as follows. Section 2 discusses related work and Section 3 introduces the concept of building blocks and the usage of hierarchical processes for a pattern-based development of self-organising systems. Finally, the last section draws a conclusion and mentions some future research goals.

2 Related Work

Self-Organisation is well known and widely spread in natural systems, e.g., swarm behaviour and neural networks, and various successful utilisations of the SO paradigm in computer systems, e.g automatically guided vehicles or ant optimisation, have been reported [7]. However, the purposeful development of self-organising systems is still a challenge since SO makes primarily use of the following concepts: autonomy, dynamics, adaptivity and decentralised organisation [3] which obviously challenges traditional top-down development approaches. Consequently, [3, 4] propose new approaches that target SO problems in which *simulation* is an inherent part of the development process. Here simulation ensures proper system behaviour and is used in *early* development stages to continuously monitor the system dynamics.

Despite these new approaches, there is still a lack of concepts that systematically support the practical usage of simulation in self-organising systems. In this regard, existing simulation support can be broadly categorised into three classes: First, there are approaches as *Swarm* or *DesmoJ* that offer simulation capabilities by utilizing standard programming languages. These frameworks are highly flexible, but often lack support for non-experienced users and do not feature a wide range of tool support. Second, there are approaches as *NetLogo* or *Arena*. These systems offer a user interface and focus more on usability and simulation support. But most of these systems are hardly customisable and applicable for different applications. Furthermore, the user of such systems is often limited to the provided functions by the system, because an integration of new methods and algorithms for the development

of modern self-organising systems is not supported. Third, there are approaches on a conceptual level that focus on specific aspects as, e.g., optimisation or validation. These approaches are often used by the self-organising community and can be seen as *process patterns* for the development of self-organising systems [3, 4, 6]. Moreover, these approaches focus on specific goals and do neither consider the generic perspective of simulation support nor do they provide general simulation-based development patterns.

Furthermore, projects as COSMOS [1] propose extensive process models to engineer from real-world systems to simulation results by suggesting different meta models to describe and analyse the system under study. In contrast to this, the concept of building blocks and hierarchical process does not focus on a specific approach to develop complex systems from scratch but rather tries to support the development of SO in general by providing a well-grounded concept to combine individual development processes. So this more functional approach can be used in such a comprehensive process model to provide the developer with reusable best-practise patterns.

3 Building Blocks for Simulation Patterns in SO

It is assumed that all tasks of the development of self-organising systems can be seen as *activities* of a coherent development process which can, according to the analysis of related work, be realised by already existing systems and concepts. Therefore, the presented approach rather tries to define a general concept that mediates between already existing solutions than providing new solutions for single development activities. The long-term objective is to allow for integrating activities of heterogeneous solutions in order to enable the development and execution of customisable, expendable and reusable patterns of any set of specified activities. These patterns can then be used to answer important questions of engineering self-organising systems, like identify macroscopic properties and variables or define steady scenarios and analysis algorithms [9]. To achieve this overall goal the concept follows three essential guidelines: (I) the set of functionalities that can rely on a external system or manually procedures should not be limited, (II) developed patterns and desired activities should be customisable and expandable and (III) all activities and processes should be reusable and interchangeable if they provide the same functionality.

Because of such a diverse domain of interest, a strict separation of activity scopes is required. In order to handle the arising heterogeneity of such separated activities the presented concept offers *building blocks* which act like an adapter for the accomplished activity and provide a distinct execution scope. These blocks operate as so called *active components* which represent a novel approach to build complex distributed applications [5]. Active components can be understood as distinct components which are managed by an infrastructure but remain autonomous in regard to their execution. The logic is realised by different internal architectures which can be based on various execution logics, e.g., agent-oriented architectures or process-based architectures. Consequently, building blocks offer the concurrent execution

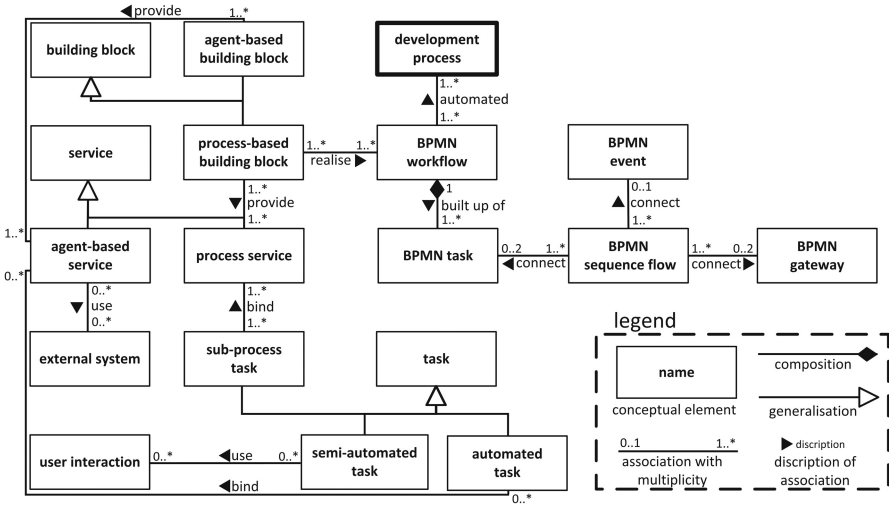


Fig. 1 Meta model for SO processes with building blocks

of self-contained actions that allows a fully independent execution, whereby any occurring dependencies of their actions remain within their building block scope.

In order to provide the functionalities and to ensure interchangeability and reusability of the activities, all functionalities of a building block are provided as one or more *services* in a service-oriented architecture which allows to publish, discover and utilise services locally or remotely. Thereby, the providing building block is responsible to interpret service requests and handle the corresponding actions in its scope which induces a loosely coupled system. For example, in case of a process-based architecture, an asynchrony service request can instantiate a new (sub-)process in which various tasks become active and perform their implemented activities which can consist of task specific program code or further service calls. The resulting system architecture of such consecutive service calls can be seen as a *Service Composition Architecture (SCA)*. This conjunction of both a process-based architecture and the use of any number of complemental services allows for complex process-based activities. In this way, a hierarchical structure of a building block, e.g., a development process, can be modelled by a service request to previously created building blocks, e.g., specialised development activities, that operate on a lower abstraction layer than the calling block. Thereby, the calling building block disguises characteristics and implementation details like, e.g., a concrete used method, by using services and, thus, enhances reusability, comprehension, and the further development of complex development processes.

Figure 1 illustrates our concept with a meta model for a combination of agent-based activities and BPMN processes. Here a development process is automated by a BPMN workflow which is built up of an arbitrary number of BPMN tasks that are connected within the modelled sequence flow. Each task can be semi-automated by

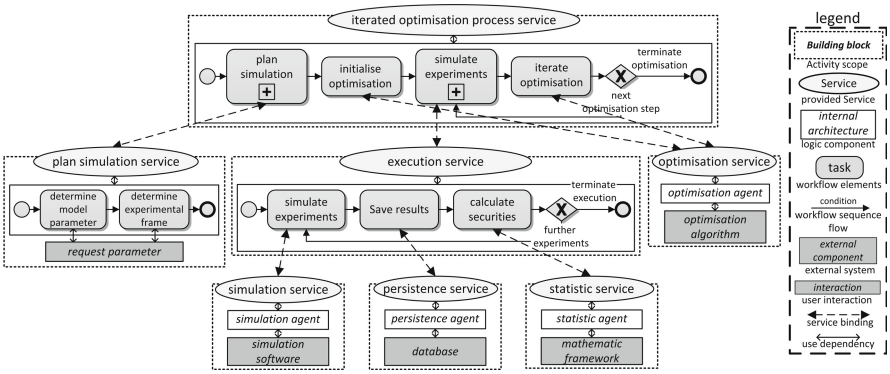


Fig. 2 A hierarchical process pattern for optimisation

a user interaction, automated by an agent-based service or can use a sub-process. Automated and sub-process activities are always provided by building blocks, that allow autonomous and independent actions and, in case of a process-based service, a hierarchical continuation by supplying a sub-process.

So on the one hand, external solutions, e.g., simulation systems or database support, can be used via service descriptions on a high abstraction level and on the other hand specialised demands can be realised by individual solutions on lower abstraction layers. As an example figure 2 shows a basic optimisation pattern. On the highest abstraction layer of this pattern a process-based building block coordinates between planing, simulation execution and an iterated optimisation algorithm. Simulation execution is realised by a simulation-based process cycle [6] which is realised within an *execution service* that executes and persists simulation runs until a defined confidence interval is reached. In doing so, single runs are performed via an agent that offers a simulation service for a specified simulation system that is integrated via a black box approach. Thereby, the presented concept also allows for specialised and encapsulated processes - rather than a strict black box model. Consequently, these processes can operate in their building block scope with knowledge about a specific system without affecting the generic usability of higher abstraction levels. This characteristic allows for model-specific runtime operation which is, for instance, rather useful in system benchmarking which is one of the most challenging tasks in SO [8]. To find an optimal solution the sequence flow of the pattern alternates between service requests to a direct optimisation method, e.g., a metaheuristic, which supplies different configurations of model parameters and the simulation execution which evaluates these configurations and provides performance indicators of the explored model. The presented example shows that prior manually performed procedures for self-organising systems can be decomposed in distinct, reusable and executable building blocks that can be merged in automatable patterns.

4 Conclusion and Future Work

This work has presented a concept that supports the systematic development of self-organising systems by reusable patterns that encapsulate activities, e.g., pure simulation, optimisation or evaluation, in *building blocks*. These encapsulated building units are addressed by a service description and can be hierarchically ordered and combined according to logical or temporal conditions. In conclusion, the presented concept can simplify the task of simulating complex systems since it allows for conveniently incorporating existing solutions by developing and executing best-practise process patterns as well as using state-of-the-art development technology. Therefore, the presented concept is highly flexible and customisable since it does not bound the developer to specific software systems and methods - a characteristic which is necessary to apply SO in real-world problems.

Future work shall, on the one hand, strive towards developing further patterns in order to support even more expedient application domains. This includes the shaping and categorisation of characteristic best-practise patterns as well as the integration of alternative development aspects. On the other hand, it is envisioned to implement an extensive simulation framework for SO that standardises and executes activities in order to allow a straightforward system design.

Acknowledgements. The authors would like to thank Deutsche Forschungsgemeinschaft for supporting this work through the project "SO based on decentralised co-ordination in distributed systems" and also W. Renz from Hamburg University of Applied Sciences for inspiring discussions.

References

1. Andrews, P., et al.: Cosmos process, models, and metamodels. In: Stepney, S., et al. (eds.) Proc. of the 2011 Works. on Complex Systems Modelling and Simulation, pp. 1–14 (2011)
2. Edmonds, B.: Using the experimental method to produce reliable self-organised systems. In: Brueckner, S. (ed.) Engi. SO Systems: Methodologies and Applications, pp. 84–99 (2004)
3. Gardelli, L., et al.: Combining simulation and formal tools for developing self-organizing MAS. In: Uhrmacher, A.U., et al. (eds.) Multi-Agent Systems: Simulation and Applications (2009)
4. Gershenson, C.: Design and control of self-organizing systems. Ph.D. thesis, Vrije Univ. (2007)
5. Pokahr, A., et al.: Unifying Agent and Component Concepts - Jadex Active Components. In: Braubach, L., et al. (eds.) 7th Ger. Conf. on MAS Technologies (MATES), pp. 100–112 (2010)
6. Robinson, S.: Automated analysis of simulation output data. In: Kuhl, M., et al. (eds.) Proc. of the 37th Conf. on Winter Simulation (WSC), pp. 763–770 (2005)
7. Sauter, J., et al.: Performance of digital pheromones for swarming vehicle control. In: Proc. of the 4th Int. Conf. on Autonom Agents & Multiagent Syst. (AAMAS), pp. 903–910 (2005)

8. Vilenica, A., Lamersdorf, W.: Benchmarking and evaluation support for self-adaptive distributed systems. In: 6th Int. Conf. on Compl., Intel. & Softw. Intensive Syst. (CISIS), pp. 20–27 (2012)
9. Wolf, T., et al.: Engineering Self-Organising Emergent Systems with Simulation-based Scientific Analysis. In: Proc. of the 4th Int. Works. on Engi. SO Applications, pp. 146–160 (2005)

Molecules of Knowledge: Self-organisation in Knowledge-Intensive Environments

Stefano Mariani and Andrea Omicini

Abstract. We propose a novel self-organising knowledge-oriented model based on *biochemical tuple spaces*, called *Molecules of Knowledge* (MoK). We introduce MoK basic entities, define its computational model, and discuss its mapping on the TuCSoN coordination model for its implementation.

1 Introduction

The issues of knowledge management are becoming critical in most of the complex computational systems of today—in particular, in socio-technical systems. There, people working in knowledge-intensive environments typically face the challenges of a huge and ever-growing collection of knowledge sources, producing possibly relevant information at a fast pace, in many heterogeneous formats – so, essentially unmanageable –, which they are anyway supposed to handle and govern in order to perform their tasks and achieve their own goals. For physicians, journalists, researchers, lawyers, even politicians – *knowledge workers*, in general – today ICT systems provide at the same time new opportunities and new obstacles: the ability to find all the relevant information needed in the short time being a issue that even the most advanced general-purpose research engines are not able to face today.

Adaptive and self-organising systems seems the only possible answer when the scale of the problem is too huge, unpredictability too high, global control unrealistic, and deterministic solutions simply do not work. Nature-inspired models provide clear examples of systems where order out of chaos is achieved by means of simple and pervasive mechanisms, based on local interactions. Also, coordination models – in particular, tuple-based ones – have already proven their effectiveness in the engineering of complex software systems, like knowledge-intensive, pervasive and self-organising ones—see [3] for a survey.

Stefano Mariani · Andrea Omicini
DISI, ALMA MATER STUDIORUM – Università di Bologna, Italy
e-mail: {s.mariani,andrea.omicini}@unibo.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 17–22.
springerlink.com

In this paper we introduce a self-organising knowledge-oriented model called *Molecules of Knowledge*. There, knowledge sources produce *atoms* of knowledge in *biochemical compartments*, which then diffuse and aggregate in *molecules* by means of *biochemical reactions*, acting locally within and between such spaces. Knowledge consumer's workspaces are mapped into such compartments, which reify information-oriented user actions in terms of their state – hence atoms and molecules in the compartments –, influencing atoms aggregation and molecule diffusion—for instance, making potentially relevant knowledge atoms and molecules autonomously move toward the interested users.

2 The *Molecules of Knowledge* Coordination Model

Molecules of Knowledge (MoK) is a biochemically-inspired model promoting self-organisation of knowledge. The main ideas behind MoK are: *(i)* knowledge should autonomously aggregate and diffuse to reach knowledge consumers, *(ii)* the biochemical metaphor fits knowledge self-organisation in distributed knowledge-intensive environments. The main abstractions of MoK are:

atoms the smallest unit of knowledge in MoK, an *atom* contains information from a *source*, and belongs to a *compartment* where it "floats";
molecules the MoK units for knowledge aggregation, *molecules* bond together related atoms;
enzymes emitted by MoK *catalysts*, *enzymes* influence MoK reactions, thus affecting the dynamics of knowledge evolution within MoK compartments;
reactions working at a given *rate*, *reactions* are the biochemical laws regulating the evolution of each MoK compartment, by governing knowledge aggregation, diffusion, and decay within MoK compartments.

Other relevant MoK abstractions are instead in charge of important aspects like topology, knowledge production, and knowledge consumption:

compartments the spatial abstraction of MoK, *compartments* provide MoK with the notions of *locality* and *neighbourhood*;
sources associated with compartments, MoK *sources* originate knowledge, which is injected in the form of MoK atoms at a certain *rate* within the compartment;
catalysts the abstraction for knowledge *prosumers*, *catalysts* emit enzymes which represent prosumer's actions, affecting knowledge dynamics within the prosumer's compartment.

2.1 Atoms

Atoms are the most elementary pieces of knowledge within the model. A MoK atom is produced by a knowledge source, and conveys a piece of information spawning from the source itself. Hence, along with the content they store – the piece of raw information –, atoms should also store some contextual information to refer to the content's origin, and to preserve its original meaning.

A MoK atom is essentially a triple of the form $atom(\ src,\ val,\ attr\ )_c$. There, *src* identifies the source of knowledge; *val* is the actual piece of knowledge carried by the atom—some content, possibly along with an associated ontology; *attr* is essentially the content's attribute, that is, the additional information that helps understanding the content—again, possibly expressed according to some well-defined ontology; c is the current *concentration* of the atom, set at injection time, then evolving according to the system evolution.

2.2 Molecules

A MoK system can be seen as a collection of atoms generated at a certain rate, wandering through compartments, and possibly colliding within the compartment they belong to. The result of collisions are the *molecules of knowledge*, that is, spontaneous, stochastic, "environment-driven" aggregations of atoms, which in principle are meant to reify some semantic relationship between atoms, thus possibly adding new knowledge to the system. Each molecule is simply a set of atoms—an unordered collection of semantically-related atoms.

A MoK molecule has then the simple structure $molecule(\ Atoms\ )_c$. There, c is the current concentration of the molecule, and $Atoms$ is the collection of all the atoms currently bonded together in the molecule—which essentially represent the "pool" of related pieces of knowledge that a chain of reactions has aggregated during the natural system evolution.

In the same way as atoms, molecules are in turn involved in (biochemical) reactions, described below: they can aggregate and generate more complex molecules based on semantical relations between some atoms of them. So, whenever it may help simplifying the view of a MoK systems, one could also see atoms as molecules with a single atom, and viceversa, so that for instance, given an atom a and a molecule $m = molecule(a)$, we can consider $a = m$ whenever useful in the model specification. Along the same line, a MoK system can also be seen as a *collection of molecules of knowledge* wandering through compartments, and reacting so as to make knowledge autonomously aggregate and diffuse from sources to prosumers.

2.3 Enzymes

One of the key features of MoK is that the system interprets prosumer's knowledge-related actions as *positive feedbacks* that increase their concentration of related atoms and molecules within the prosumer's compartment. To this end, the MoK model includes *catalysts* (representing prosumers, typically knowledge workers) injecting *enzymes* whenever they access atoms and molecules in their own *compartment*. Enzymes are then used by MoK reactions to increase involved atoms/molecules' concentration, producing the positive feedback required to enable self-organisation of knowledge.

A MoK enzyme has the structure $enzyme(\ Atoms\ )_c$. There, an enzyme – with its own concentration c – explicitly refers to a collection of $Atoms$ that the catalyst's

actions have in any way pointed out as of interest—either explicitly, by an epistemic action, or implicitly, by a generic knowledge-related action.

2.4 Reactions

The behaviour of a MoK system is actually determined by the last fundamental abstraction of the model, the one actually defining the behaviour of a MoK system: the *(biochemical) reaction*. Biochemical reactions drive atoms and molecules aggregations, as well as atoms and molecules reinforcement, decay, and diffusion.

As a knowledge-oriented model, the main issue of MoK is determining the semantic correlation between MoK atoms. So, in order to define a MoK system, one should first of all define the basic *mok* function, taking two atoms — as in $mok(\ atom_1,\ atom_2\ )$ – and returning true if $atom_1$ and $atom_2$ are semantically related. The precise definition of *mok* depends on the specific application domain: different notions of semantic correlation could be used depending on the sort of the knowledge-intensive environment—such as news, scientific research, health care. On the other hand, it requires the analysis of the possible correlations between the ingredients of the atoms, so that the value of $mok(atom_1, atom_2)$ typically depends on the *val* and $attr$ elements of $atom_1$ and $atom_2$.

The *aggregation reaction* bounds molecules based on semantic correlation:

$$molecule(Atoms_1)\ +\ molecule(Atoms_2) \longmapsto^{r\_agg} molecule(Atoms_1 \bigcup Atoms_2)\ +\ Residual(Atoms_1, Atoms_2)$$

There, *r_agg* is the reaction rate; $mok(atom_1, atom_2)$ holds for some $atom_1 \in Atoms_1$, $atom_2 \in Atoms_2$; and $Residual(Atoms_1, Atoms_2)$ is the multiset of atoms obtained as the difference $(Atoms_1 \uplus Atoms_2) \setminus (Atoms_1 \bigcup Atoms_2)$—in short, all the atoms of $Atoms_1$ and $Atoms_2$ that do not belong to the resulting molecule, thus preserving the total number of atoms. In short, more complex molecules are formed by aggregation whenever some atoms in the reacting molecules (or in reacting atoms, handled here as one atom molecules) are semantically correlated (via the *mok* function).

Positive feedback is obtained by the *reinforcement reaction*, consuming an enzyme injected by a catalyst to produce a single unit of the relevant atom/molecule:

$$enzyme(Atoms_1)\ +\ molecule(Atoms_2)_c \longmapsto^{r\_reinf}\ molecule(Atoms_2)_{c+1}$$

There, *r_reinf* is the reaction rate; $enzyme(Atoms_1)$, $molecule(Atoms_2)_c$ exist in the compartment, with $c \neq 0$; and $mok(atom_1, atom_2)$ holds for some $atom_1 \in Atoms_1$, $atom_2 \in Atoms_2$.

Some MoK biochemical reactions are meant to make knowledge evolve according to the space and time patterns that typically characterise self-organising scenarios. So, in order to provide the required *negative feedback*, molecules should fade as time passes, lowering their own concentration according to some well-defined decay law. The temporal *decay reaction* is hence defined as follows:

$$molecule(Atoms)_c \longmapsto^{r\_decay}\ molecule(Atoms)_{c-1}$$

where one molecule instance disappears, at a rate *r_decay*, hence its concentration is decreased by one.

Analogously, a distributed self-organisation model should provide some kind of spatial evolution pattern. According to its natural inspiration, MoK adopts *diffusion* as its knowledge migration mechanism. Here, diffusion is simply the migration of an atom or molecule from one chemical compartment to another, thus providing the basic mechanism for knowledge sharing and autonomous motion in a distributed system. According to the biochemical metaphor, atoms and molecules can only diffuse between *neighbour* compartments, resembling membrane crossing among cells. Assuming that σ identifies a biochemical compartment, and $\|\|_\sigma$ brackets molecules in a compartment σ, the *diffusion reaction* is modelled as follows:

$$\begin{array}{c} \| \ Molecules_1 \bigcup molecule_1 \ \|_{\sigma^i} + \| \ Molecules_2 \ \|_{\sigma^{ii}} \longmapsto^{r\_diffusion} \\ \| \ Molecules_1 \ \|_{\sigma^i} + \| \ Molecules_2 \bigcup molecule_1 \ \|_{\sigma^{ii}} \end{array}$$

There, σ^i and σ^{ii} are neighbour compartments, *r_diffusion* is the diffusion rate, and $molecule_1$ moves from σ^i to σ^{ii}. Along with reinforcement and decay, diffusion ensures that relevant knowledge is brought toward interested prosumers: roughly speaking, diffusion scatters knowledge around, reinforcement increases the concentration of relevant knowledge, and decay abates non-relevant one.

3 Mapping MoK over TuCSoN

TuCSoN [4] is a coordination model & infrastructure exploiting programmable tuple spaces, called *tuple centres*. The behaviour of a tuple centre can be defined through the ReSpecT logic language [2] to encapsulate any coordination law into the coordination abstraction—so, in principle, biochemical reactions, too.

Since logic-based ReSpecT tuple centres are well-suited to handle knowledge representation in general, our first implementation of MoK is built upon TuCSoN. In particular, TuCSoN easily provides the two basic ingredients for biochemical coordination: *(i)* chemical-inspired stochastic evolution of tuples – achieved by implementing the well-known Gillespie's exact simulation algorithm [1] as a ReSpecT specification –, and *(ii)* matching of biochemical laws against tuples in the space—obtained by interpreting tuples as individuals in the biochemical system, and using templates to denote reactants in biochemical reactions.

According to the MoK biochemical metaphor discussed in Section 2, the first natural mapping of MoK concepts over TuCSoN abstractions is the following:

Individuals → Atoms / Molecules / Enzymes → (Logic) Tuples

Viewing computational systems as eco-systems, individuals are the inhabitants of the environment, ruled by its laws, and at the same pro-actively affecting the environment itself. MoK individuals are all the knowledge chunks floating in a biochemical compartment—the subjects and enablers of its biochemical reactions. In TuCSoN all the abovementioned abstractions translate into (logic) tuples.

The representation of the environmental substrate in terms of TuCSoN tuple centres is plain: the environment stores both the individuals and the laws of nature in the same way as tuple centres store tuples and coordination laws:

Environmental Substrate → *Compartments / Catalysts* → TuCSoN *Tuple Centres*

Since catalysts are users, and each user has a compartment as its working place, both catalysts and compartments are in principle mapped upon tuple centres.

Finally, in a biochemical compartment, the role of the laws of nature is played by biochemical reactions, built as ReSpecT reactions:

Laws of Nature → *Biochemical Reactions* → ReSpecT *Reactions*

Each TuCSoN tuple centre representing a MoK compartment is programmed to work as a *biochemical virtual machine* implementing the Gillespie's chemical simulation algorithm, enacting MoK biochemical reactions as ReSpecT reactions.

Based on the above mapping, the prototype implementation of MoK has been already developed, then tested on a first set of experiments—not reported here, however, for the lack of space.

Acknowledgements. This work has been supported by the EU-FP7-FET Proactive project SAPERE Self-aware Pervasive Service Ecosystems, under contract no.256873.

References

1. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The Journal of Physical Chemistry 81(25), 2340–2361 (1977), doi:10.1021/j100540a008
2. Omicini, A., Denti, E.: From tuple spaces to tuple centres. Science of Computer Programming 41(3), 277–294 (2001), doi:10.1016/S0167-6423(01)00011-9
3. Omicini, A., Viroli, M.: Coordination models and languages: From parallel computing to self-organisation. The Knowledge Engineering Review 26(1), 53–59 (2011), doi:10.1017/S026988891000041X
4. Omicini, A., Zambonelli, F.: Coordination for Internet application development. Autonomous Agents and Multi-Agent Systems 2(3), 251–269 (1999), doi:10.1023/A:1010060322135

Emergent Distributed Bio-organization: A Framework for Achieving Emergent Properties in Unstructured Distributed Systems

George Eleftherakis, Ognen Paunovski,
Konstantinos Rousis, and Anthony J. Cowling

Abstract. Distributed systems are particularly well suited to hosting emergent phenomena, especially when individual nodes possess a high degree of autonomy and the overall control tends to be decentralized. Introducing novel bio-inspired behaviours and interactions among individual nodes and the environment as a means of engineering desirable behaviours, could greatly assist with managing the complexity inherent to artificial distributed systems. The paper details the Emergent Distributed Bio-Organization (EDBO) as an abstract distributed system model aiming to engineer emergent properties at the macroscopic level. EDBO was designed to be suitable as a starting point in the design of a specific class of problems of real-world distributed systems. A thorough discussion justifies why the proposed bio-inspired properties planted into the model, could potentially allow for the desired behaviours to emerge.

1 Introduction

The last decades have experienced an increased application of distributed solutions to information systems, spanning a diverse set of domains with growing and more demanding user base. The ever increasing complexity requires worrisomely more

George Eleftherakis
The University of Sheffield International Faculty, CITY College, 3, Leontos Sofou Str., 54626 Thessaloniki, Greece
e-mail: eleftherakis@city.academic.gr

Ognen Paunovski · Konstantinos Rousis
South-East European Research Centre (SEERC), 24, Proxenou Koromila Str., 54622, Thessaloniki, Greece
e-mail: {ogpaunovski,konrousis}@seerc.org

Anthony J. Cowling
The University of Sheffield, 211 Portobello, Sheffield, UK
e-mail: A.Cowling@dcs.shef.ac.uk

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 23–28.
springerlink.com

human resources and skills to tackle the various aspects of such complex system lifecycles. In this context, a number of novel approaches have used biological systems as inspiration in the design of artificial distributed systems (ADS), aiming for solutions to various problems and challenges encountered. The rationale for using biological concepts and processes to the design and operation of distributed systems is based on the fact that the biological distributed systems have already created elegant solutions that utilize reactive micro-level behaviours and interactions to give rise to emergent system-wide properties like: availability, adaptability, scalability, self-organization, and other self-* properties [2, 5, 6].

The Emergent Distributed Bio-Organization (EDBO) model is an outcome of ongoing work on engineering ADS which could harness emergence. Previous work on exploring emergence and understanding the micro-macro causal links [3], along with experience obtained in the field of ADS, with a simpler version of EDBO [4], has led to the development of a disciplined framework for engineering emergence [1]. This paper details EDBO as a model conceived by following this framework, and discusses the hypotheses formulated in order to allow the emergence of desired global properties. Typically, bio-inspired approaches are focused on solving a very specific problem with limited scope while this work assumes a more holistic approach by addressing multiple issues in the operation of distributed systems. EDBO was designed as a generic case study based on principles common to many unstructured ADS, allowing the expected solutions to be easily adapted to various real-world systems. Nonetheless, it serves as a case study that utilizes simulation to demonstrate how global emergent properties can be engineered by focusing on the microscopic level.

The rationale of introducing emergent properties in a distributed system is to improve its operational efficiency and to eliminate the need for human configuration and management. The main problem that EDBO attempts to solve is the discovery of resources in an unstructured, fully decentralized, network, under varying conditions. EDBO is a continuation of previous work in distributed systems where the focus was on understanding how relationships among network nodes affect resource discovery overall [4]. Although it is a generic case study representing distributed systems in an abstract way, specific case studies such as file sharing networks, high performance computing, or decentralized schemes of Web services, could potentially benefit from the results of this work.

While there is a diverse set of objectives that contemporary distributed systems are expected to satisfy, not all of them can be appropriately treated as macroscopic properties which could emerge from local interactions. The main objectives of EDBO are to achieve scalability, robustness and availability in a distributed system without explicit engineering. Instead, by following the proposed experiment-driven framework, various properties and behaviours are introduced as different hypotheses of achieving the aforementioned global objectives. Emergent properties can be utilized in the organization and maintenance of the network as well as the basis for improving the discovery of resources within the network. The next section details the EDBO model and the incorporated bio-inspired concepts and operational principles that could lead to the desired emergent behaviour.

2 Emergent Distributed Bio-organization

The scope of the EDBO study and the corresponding model is limited to the top-most layer of the distributed systems archetype. The main goal of the study is to devise a generic unstructured distributed system model which will maintain a high level of availability, scalability, and robustness, under different operational conditions. In order to achieve this goal, EDBO introduces biologically inspired behaviours in combination with concepts from agent based modelling as a foundation for achieving emergent properties in an artificial distributed system. The conceptual structure followed in the model is built in terms of a distributed application that provides resources to external users, rather than a distributed system that is composed of users (peers).

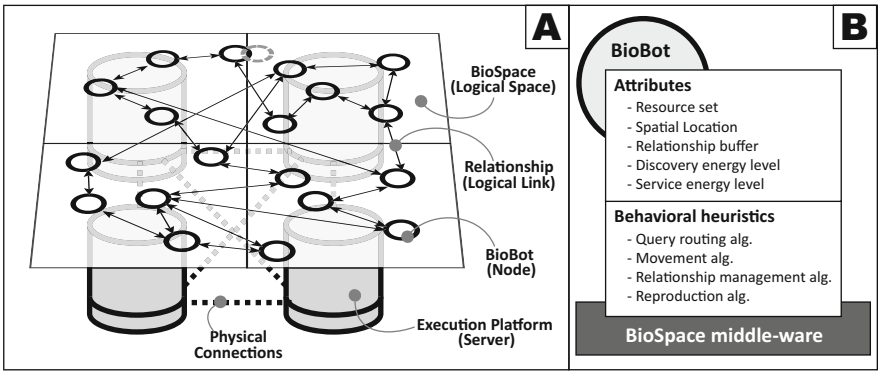


Fig. 1 A - Simplified representation of the BioSpace; B - BioBot's main attributes and heuristics

Nodes in the EDBO model are represented by agent-like entities referred to as *BioBots* (fig. 1-B) which use two-way logical connections (relationships) to form the overlay network. Each BioBot has a limited number of relationships to other BioBots which are managed autonomously. In terms of the distributed application, a BioBot serves as a wrapper for a set of resources (abstracting data, functionality and services) that are provided to the requests (queries) issued by the users. Moreover, a BioBot represents the main autonomous routing point which facilitates the propagation of queries on the network. The operation of the BioBot is based on several bio-inspired heuristic (elaborated in 2.1) mechanisms that guide its decision making. The heuristics rely on the BioBot's internal state, the available relationship meta-data, and the state of the environment which in terms of the EDBO paradigm is referred to as *BioSpace*. The BioSpace acts as a middleware between the BioBots and the underlying operating system and physical infrastructure (see fig. 1-A), with different middleware instances running on different servers being interconnected in order to facilitate a unified environment and a singular spatial universe. In addition,

the BioSpace facilitates several functional operations such as the communication service, the offspring creation service, the movement service, and others.

2.1 EDBO - Biological Principles

The EDBO platform incorporates several bio-inspired concepts that form the basis for achieving global optimization and organization from the individual bio-behaviours incorporated into the BioBot.

Autonomous decision-making - Autonomy is one of the fundamental principles of nature; biological entities usually act without explicit central or external control. In the EDBO model the concept of autonomous decision making is realized through several algorithms that use local interaction and local information to guide the BioBot's behaviour.

Death and birth events through energy maximisation - Energy can be viewed as a major life force in the natural world that allows biological entities to perform tasks. In the EDBO model the concept of energy is incorporated as an attribute of the BioBot. The model proposes two different types of energies:

- *Discovery energy* is the primary indicator of BioBots' usefulness in facilitating the resource discovery process. The discovery energy is awarded to the BioBots that facilitate a successful query match. BioBots may spend discovery energy for a variety of activities, including message exchange, movement, creation of offspring, and others.
- *Service energy* denotes the usefulness of a resource in the system. Service energy is allocated to each resource individually when the resource matches a query. Service energy is consumed each time query matching occurs.

In the natural world, biological organisms aspire to maximize the energy gain and minimise the energy loss as a vital principle for survival. The same principle of energy maximisation is utilized in the EDBO. BioBots that deplete their energy die (are removed from the system), while bots that have high energy levels are able to create offspring. The creation of a new BioBot can be realized either through replication, which results in an identical copy of the parent, or reproduction where properties of two BioBots (discovery parent and service parent) are combined.

Adaptation through ad-hoc selection - The idea of ad-hoc node selection in EDBO is inspired by the basic principle of natural selection and fitness-based evolution in the natural world. Fitness in EDBO is expressed through the energy levels of BioBots, forming thus the basis for differentiation between fit and unfit entities. Through death and birth events, the model should be able to continuously select the most successful BioBots with respect to the current network conditions.

2.2 EDBO - Operational Emergence

The central issue in the EDBO study is to devise effective and efficient means for discovery of resources in an initially unstructured, fully decentralized, network

which uses simple keyword comparison for the purpose of query matching. Although query routing seems to be the main factor for the success of the resource discovery process, we believe that providing the right network organization and resource population size is of equal importance. The rationale behind this is that by imposing ad-hoc overlay organization and adjustment of the number of BioBot/resource instances, the system will be able to maintain acceptable levels of query match success rate without proactive (well-informed) routing strategies that often impose high communication overheads. Towards this direction, the EDBO platform aims to utilize emergent behaviours in two main aspects of operation: the availability and scalability of resources and the organization and optimization of the network.

The scalability and availability of resources is an important aspect in ADS, since the query load imposed by the users often exhibits extreme fluctuations. Additionally, some of the resources are in higher demand (popular) than others (unpopular). In order to cope with these requirements, EDBO utilizes energy distribution as the main feedback mechanism that regulates the size of the BioBot population. Since the amount of energy entering the system is regulated by the number of successfully resolved queries, in situations where there is a high query load the amount of available energy in the system will be increased. This allows for a larger BioBot reproduction rate which essentially scales up the number of available resource providers. Moreover, by using energy driven selection of parents (see 2.1) popular resources have higher chances of being reproduced. A decrease in the query load leads to an overall energy decrease, which in turn increases the Biobots' death rate and reduces the number of birth events. Thus, the size of the BioBot population scales down.

Unstructured overlays commonly fail to produce the resource discovery performance of their structured counterparts. EDBO addresses this issue through emergent overlay organization and connection optimization. The main inspiration behind that is the super-node architecture that enables application of discovery mechanisms which improve overall discovery performance [7]. In the super-node architecture, the overlay structure can be logically divided into two layers: the global layer which is composed out of super-nodes that enable global connectivity and the local cluster layer which facilitates local interconnectivity. In order to facilitate the ad-hoc formation of super-node structures through local BioBot decisions, EDBO incorporates flexible relationship buffer size, different relationship acquisition strategies, and energy meta-data. The selection of the super-bots is based on the BioBots' discovery energy level, which in turn determines the number of relationships (slots) that a BioBot can establish with others. BioBots with high discovery energy and high number of relationships are granted the status of super-bots. Super-bots allocate part of their connections to other super-bots, while the rest of the connection slots are used to facilitate the connectivity with the nodes in the local cluster. BioBots in the local cluster optimize their relationships through evaluation of partners' spatial distance, thus ensuring tight connectivity in the local cluster.

3 Conclusion and Future Work

Biological distributed systems seem to effortlessly create elegant and efficient solutions to many challenges common for ICT systems that classical engineering approaches have failed to achieve. This work utilizes inspiration from nature by harnessing concepts and principles used in the biological world to form a template for achieving emergent properties in artificial systems. The proposed EDBO platform represents a continuation of earlier work in the area of unstructured distributed networks [4]. However, it substantially extends the scope of the initial study in order to achieve scalability, availability and adaptability through macroscopic organization and optimization which emerges from microscopic behaviours and interactions.

Ongoing work is concentrated on performing an experiment-driven evaluation of the EDBO model through multi-agent simulation. Iterative revision and refinement of the behaviour and interactions in the model is performed in order to devise appropriate heuristics that would achieve the desired emergent properties. Future work will address the issue of finding the right balance between the properties of the model, as a key element for developing a holistic emergent solution for a distributed application without predetermined structural design. In addition, other aspects of the ADS operation will be explored such as the development of a discovery algorithm which could adapt and benefit from the emergent overlay organization.

References

1. Eleftherakis, G., Paunovski, O., Rousis, K., Cowling, A.J.: Harnessing emergent properties in artificial distributed networks: an experimental framework. In: Ritson, C., Andrews, P., Stepney, S. (eds.) 4th Workshop on Complex Systems Modelling and Simulation, pp. 141–144 (2011)
2. Nakano, T., Suda, T.: Applying biological principles to designs of network services. Applied Soft Computing 7, 870–878 (2007)
3. Paunovski, O., Eleftherakis, G., Cowling, A.: Disciplined Exploration of Emergence using Multi-Agent Simulation Framework. Computing and Informatics 28(3), 369–391 (2009)
4. Paunovski, O., Eleftherakis, G., Dimopoulos, K., Cowling, A.: Evaluation of a selective distributed discovery strategy in a fully decentralized biologically inspired environment. Information Sciences 180(10), 1865–1875 (2010)
5. Stepney, S., Polack, F., Turner, H.: Engineering emergence. In: Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems, pp. 89–97 (2006)
6. Suda, T., Nakano, T., Moore, M., Enomoto, A., Fujii, K.: Biologically Inspired Approaches to Networks: The Bio-Networking Architecture and the Molecular Communication. In: Liò, P., Yoneki, E., Crowcroft, J., Verma, D.C. (eds.) BIOWIRE 2007. LNCS, vol. 5151, pp. 241–254. Springer, Heidelberg (2008)
7. Yang, B., Garcia-Molina, H.: Designing a super-peer network. In: IEEE International Conference on Data Engineering, ICDE 2003 (2003)

A Self-organized Approach for Detecting Communities in Networks

Ben Collingsworth and Ronaldo Menezes

Abstract. Community structure reveals hidden information about networks which cannot be discerned using other topological properties. The prevalent community detection algorithms, such as the Girvan-Newman algorithm, utilize a centralized approach which offers poor performance. We propose a self-organized approach to community detection which utilizes the newly introduced concept of node entropy to allow individual nodes to make decentralized decisions concerning the community to which they belong. Node entropy is a mathematical expression of an individual node's satisfaction with its current community. As nodes become more "satisfied", i.e., entropy is low, the community structure of a network is emergent. Our algorithm offers several advantages over existing algorithms including near-linear performance, identification of community overlaps, and localized management of dynamic changes in the network.

1 Introduction

Community structure reveals hidden information about networks which cannot be discerned using other topological properties. Early research in this area focused on a limited aspect of this problem, reasonably, finding communities. However, these solutions neglected important properties when dealing with real-world networks: *(i)* the enormous size of many networks, *(ii)* community overlap (i.e., a node belonging to multiple communities), and *(iii)* the dynamic nature of networks. The importance of the first property (network size) is evident when running one of the various network analysis packages on a very large network; the package may require many hours to run. This issue is heightened with increased interest in complex network

Ben Collingsworth · Ronaldo Menezes
Florida Institute of Technology, Computer Sciences
BioComplex Laboratory
Melbourne, Florida, USA
e-mail: bcolling@my.fit.edu, rmenezes@cs.fit.edu

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 29–39.
springerlink.com

analysis coupled with accessibility to databases capable of generating huge networks. The second property (community overlap) is the reality that nodes in a network often belong to more than one community. This property is particularly evident in social networks describing direct relationships between people. For example, in a network describing ties between students, an individual may participate in activities such as academic societies, sports activities, and religious affiliations, which may constitute several communities. Important information is lost if this student must be cast into a single community. Finally, real-world networks are constantly evolving, with nodes and edges dynamically being created and removed. Networks representing the spread of pandemic diseases are a good example of these dynamic structures. Here, the network is continuously changing as the disease spreads to new individuals and expires in those that are infected. In these networks, a community detection algorithm that must be completely re-executed with each update is impractical.

We propose a community detection algorithm which observes these three properties while maintaining partition quality. In this algorithm, individual nodes are independently responsible for determining the community to which they belong. The mechanism for making this decision is derived from Shannon entropy [13] and requires a node to have knowledge only of its immediate neighbors. Initially, entropy is high, and there is tension in the network as communities form and nodes make decisions to join or leave these communities. Over time, entropy becomes low as nodes are satisfied with their current community. At this point, community structure is emergent. Since each node's decision on community is based only on immediate neighbors, the algorithm offers linear performance on average. Additionally, near-linear performance is preserved as the size of the network increases. Nodes that belong to multiple communities are identified by a high individual entropy when the overall network has stabilized. These nodes may be seen to toggle between communities because they are uncertain about where they belong. Finally, dynamic changes to the network are processed locally. The processing required to adapt to a change in the network is proportional to the number of nodes directly impacted by the change.

2 Related Work

In this section, three algorithms are described which are representative of these categories. Each description includes a brief evaluation in terms of the characteristics described in Section 1: *(i)* complexity, *(ii)* community overlap detection, and *(iii)* adaptability to dynamic network changes.

A broad class of community detection algorithms may be termed metric-based heuristic algorithms where some network, node, or edge property is calculated across the network by means of a centralized control process, and used to partition the network into communities. A well-known example of this type of algorithm is the Girvan-Newman [6] algorithm. This algorithm utilizes edge centrality to perform a sequence of divisive cuts in the network which results in a community partition. Intuitively, the edge with the highest betweenness is likely to be a bridge

between communities, and its removal will result in the isolation of two communities previously connected by it. The algorithm is run iteratively, where at each iteration, an edge is selected and removed. Additionally, each iteration updates a dendrogram structure which reflects the current partition. The algorithm is terminated when all edges have been removed. At this point, the dendrogram is used to select the desired level of community decomposition. The performance of the algorithm is obviously tied to the edge betweenness calculation which is $O(n^2)$ for each iteration which leads to a complexity of $O(n^2m)$, where m is the number of iterations (edges) and n is the number of nodes in the network. Optimization techniques have been developed to reduce the cost of this calculation. However, these optimizations impact the quality of the partition. The algorithm does not reveal community overlap; each node is assigned to a single community. Further, the algorithm is not suited for dynamic networks since changes in network topology require a completely new dendrogram to be needed (restart of the who process).

Another large class of community detection algorithms utilizes spectral analysis to partition networks. These algorithms, referred to as spectral clustering algorithms [4], transform the relationship between nodes into a spatial relationship in which similarities between nodes are much more evident, thereby simplifying community detection. The transformation is done by generating and evaluating the eigenvectors of various matrixes derived from the network adjacency matrix. Laplacian matrixes are particularly effective for this transformation because the eigenvectors produced correspond to node coordinate vectors in k-dimensional space, where nodes belonging to the same community are in close proximity to each other. From this transformation, simple clustering algorithms such as k-means may be applied to identify the set of communities, based on node density in the spatial representation of the network. An early example of the use of spectral clustering is given by Shi *et al.* [14]. Here, the algorithm is used to accomplish perceptual grouping (i.e., image partitioning). They reduce the problem to simple network partitioning by transforming images into spatial representations based on the attributes such as brightness, color, texture. The algorithm groups image features into communities by examining the coherence of these attributes in Euclidean space. Similar to the Girvan-Newman algorithm, spectral clustering generates good community partitions, but has complexity issues. The complexity of producing the eigenvectors for a network containing n nodes is $O(n^3)$. Again, strategies have been developed to reduce this complexity, but at the cost of partition quality. Given that spectral clustering reveals each node's proximity to other nodes, and hence proximity to communities, community overlap detection is achievable. Indeed, Ma *et al.* [8] have demonstrated this capability by extending the traditional spectral clustering algorithm to recognize candidate overlapping nodes through examination of spatial density. Finally, as shown by Ning *et al.* [11], spectral clustering may be applied to dynamic networks. This is accomplished by maintaining incremental approximations of the eigenvalues and eigenvectors rather than performing a recalculation of eigenvectors after each update to the network. This approach has shown promising initial results. However, Ning *et al.* concede that the incremental approximations incur cumulative errors, which impact the algorithm's performance.

Many hybrid solutions have been developed to address the community detection problem. One interesting example of this is the solution proposed by Cruz *et al.* [3]. They apply a combination of algorithms for community detection in social networks. First, the agglomerative Blondel [2] algorithm is used to perform community partitioning based on network structure. Blondel's algorithm discerns community structure by grouping nodes in the configuration which maximizes Girvan-Newman modularity [10]. Following the partitioning by Blondel's algorithm, community structure is enhanced through observation of semantic information contained in the network. The semantic information consists of attributes assigned to each node. For example, in a network of employees, the attributes might include age, gender, and profession. An entropy calculation is used to measure the level of similarity between a node and the peers in its community. The goal of entropy reduction is achieved by means of Monte Carlo selection. In this process, nodes are selected and moved to the community where the lowest entropy value is calculated. The combination of algorithms are executed iteratively, maximizing modularity with Blondel's algorithm and minimizing entropy through the relocation of nodes to communities with common attributes. Since modularity optimization increases entropy, and entropy optimization decreases modularity, a balance must be struck between the two algorithms. The overall process favors modularity by terminating when modularity improvements are no longer achievable. While the Blondel partitioning algorithm offers a complexity of $O(n)$, it has been shown to have accuracy issues which lead to spurious partitions during hierarchical agglomeration [4]. In addition, Girvan-Newman Modularity exhibits a resolution loss when applied to dense networks [5]. The use of semantic information improves partition quality. However, the overall approach of Cruz *et al.* does not support community overlap or dynamic networks. The use of semantic information suggests a capability for community overlap detection. However, the semantics have the effect of grouping nodes that have the highest commonality among the entire set of semantics. For example, using the student network previously described, students belonging to same academic society, sports team, and religious group would be attracted to each other to form a single community rather than each individual being assigned to four overlapping communities.

3 Algorithm Description

The algorithm is inspired by self-organized systems observed in nature. In these systems, complex goals are achieved with high efficiency and robustness [9]. The bottlenecks associated with centralized control are removed as work is performed by individuals possessing a minimal amount of intelligence and tools required for the task. From the apparent chaos of no centralized leadership, highly efficient processes are deployed with frequently astounding emergent results.

The basis for the self-organized community detection algorithm is node entropy calculation. Node entropy is derived from Shannon entropy, and expresses the certainty each node has with regard to its current community. The use of entropy is

consistent with the simple localized algorithms employed by other self-organized systems. The work of discerning community structure is focused on individual nodes rather than entire network. Equation 1 shows the node entropy (S) calculation.

$$S = -\sum_{i=1}^{n} \frac{p_i \log_2 p_i}{e^{\frac{k}{4}}}, \tag{1}$$

where, n is the number communities a node may potentially join (including its current community), and k is the number of intra-community triads formed by joining a particular community. As shown by the equation, intra-community triad structures are favored by the algorithm. Knowledge beyond a node's immediate neighborhood is not required to perform the calculation. To determine which community a node belongs to, the entropy calculation is performed with the node temporarily assigned to each community it could possibly join (p_i). The resulting entropy values are then evaluated to make a decision on community assignment. For example, Figure 1 shows a simple network where node 4 must decide which community to choose.

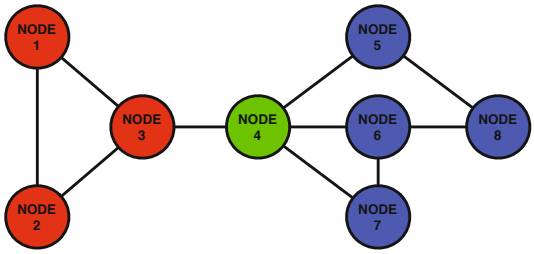


Fig. 1 Simple example network demonstrating node entropy calculation

The entropy calculation is done for each of the three choices node 4 has:

1. Stay in the green community:
   $S_g = -(\frac{1}{5}log_2\frac{1}{5} + \frac{3}{5}log_2\frac{3}{5} + \frac{1}{5}log_2\frac{1}{5}) = 1.37$
2. Join the blue community:
   $S_b = -(\frac{4}{5}log_2\frac{4}{5} + \frac{\frac{1}{5}log_2\frac{1}{5}}{e^{.25}}) = 0.55$
3. Join the red community:
   $S_r = -(\frac{2}{5}log_2\frac{2}{5} + \frac{3}{5}log_2\frac{3}{5}) = 0.972$

To make the decision, the entropy values are first inverted and normalized. Following this, a roulette wheel selection is used to select a community.

Algorithm 1 describes the overall community detection process. It executes iteratively until the average entropy of the network stabilizes to an emergent community

Algorithm 1. Community Detection

```
Ensure: each node initially assigned to unique community
  while last 3 deltas in avg network entropy > 0.1% do
    compile list of nodes
    for each node do
      randomly choose node
      if all node neighbors are not in the same community then
        calculate entropies for joining each community
        invert and normalize entropies
        perform roulette wheel selection of community
        node joins selected community
      end if
      remove node from list
    end for
  end while
```

structure. During each iteration, a growing subset of nodes is identified which are completely satisfied with their current community; that is, all of their neighbors are in the same community as themselves. For these nodes, the entropy calculations are not performed. An added performance characteristic of the algorithm is the ability to calculate the node entropy values required for each iteration in parallel. As may be observed in any beehive or anthill, concurrent activity is the rule for self-organizing systems. With today's multi-core and parallel computer architectures, algorithms which support concurrent operations enjoy a significant performance advantage over centralized algorithms.

4 Experimental Results

This section examines the algorithm's performance in terms of the three metrics described in Section 1: complexity, community overlap detection, and adaptability to network changes. In addition, since the algorithm is newly introduced, partition quality is examined. The Zachary's Karate Club Network [15] is used to evaluate partition quality. Although it is relatively small with 34 nodes and 77 edges, this network has become a standard for evaluating community detection algorithms because, through participant feedback, the real-world community structure is known. Additionally, the network contains subtle topological features which make partitioning non-trivial. The network represents a karate club which split into two factions after a dispute between the club president and the lead instructor. Figure 2 shows the community structure discerned by the algorithm.

The algorithm correctly partitions the network between the club president (node 34) and the instructor (node 1). Further, high resolution is demonstrated by the correct identification of two sub-communities within this main division.

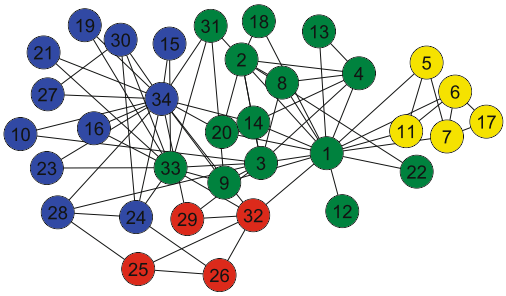


Fig. 2 Partitioned Zachary's Karate Club Network

The complexity of the algorithm is evaluated using Kleinberg small-world networks [7] generated by means of the Gephi Network Analysis Package [1]. The generated networks are quite dense, with topologies that do not easily yield community structure. Ten networks were generated, with the smallest network containing 1,000 nodes and the largest containing 10,000 nodes. In the networks between the smallest and largest, 1000 nodes were added incrementally. Each node in the generated networks has a degree of four. Figure 3 shows the number of iterations required to partition each of these networks.

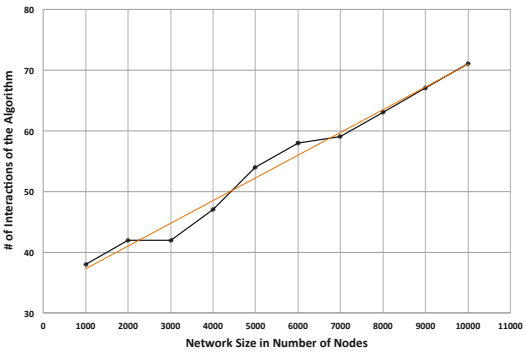


Fig. 3 Algorithm iterations required for a variety of network sizes

In Figure 3, a linear progression is seen as the networks become more complex. An additional performance characteristic may be seen in Figure 4, where the number of entropy calculations performed decreases as the algorithm progresses.

This reduction in entropy calculations is attributed to the emergent community structure. As communities are formed, an increasing number of individual nodes

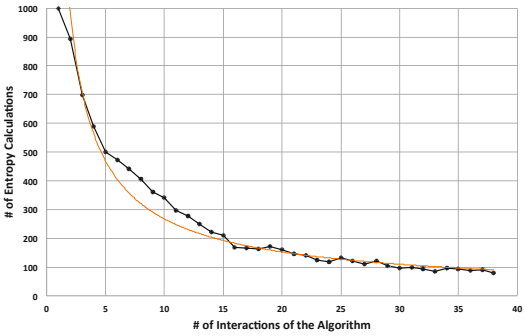


Fig. 4 The number of node entropy calculations required with each iteration of the algorithm

are "satisfied" with their community and no entropy calculation is required. As a result, in the last twelve iterations of the algorithm, entropy is calculated for only 10 percent of the nodes.

Community overlap may be discerned from the node entropy values seen in a partitioned network. The final entropy values of four nodes with interesting entropies from the karate network are listed below:

- *Node 2*: 0.0
- *Node 5*: 0.36
- *Node 6*: 0.107
- *Node 29*: 1.0

These entropies are an indication of the uncertainty of each node with its current community. Here, a higher value indicates higher uncertainty. Nodes with an entropy value of zero, such as node 2, are surrounded by neighboring nodes in the same community. Hence, there is no uncertainty concerning which community these nodes belong in. Nodes with high entropy are identified as nodes which overlap into multiple communities. For example, node 29, has an entropy value of 1.0. This entropy value indicates a 100 percent chance that node 29 will re-evaluate its community position. Node 29 has a single link to three separate communities (green, red, and blue), and no intra-community triads are formed if the node were to join any one of these communities. With each iteration of the algorithm, the entropy value which results from the node joining each of these communities is calculated. These three entropies are equal, with a value 1.0. As a result, the node has equal probability of selecting any one of these communities. Indeed, during the final iterations of the algorithm, this node is seen to toggle between the three communities. Node 5 has a final entropy of 0.36. It has one edge going into the green community, and two edges going into the yellow community with no intra-community triads. Because of the higher number of edges in the yellow community, its entropy favors remaining in the yellow community. Node 6 has an entropy of 0.107 which corresponds to having 3 edges in

the yellow community, one edge in the green community, and an intra-community triad in the yellow community. From the entropies calculated which correspond to the communities a node may belong to, a ratio of the "belongingness" of the node to each community may be determined. This concept corresponds to fuzzy community structures described by Reichardt *et al.*, where the robustness of node assignments to communities is examined to obtain a higher resolution of community structure than seen by traditional algorithms which detect overlap [12]. For example, in the karate network, node 29 belongs equally in the green, red, and blue communities. In general, thresholds may be used to determine whether a community has sufficient node attraction to indicate overlap. Initially, overlap may be detected by a high final node entropy value. Following this, the level of participation of the node in each community may be discerned from the entropy values calculated when the node was evaluated in each community during the last iteration of the algorithm. Here, a community may be eliminated as an overlap candidate if the node entropy is sufficiently high when evaluated in that community.

The amount of processing required by the algorithm to incorporate real-time updates to a network is proportional to the complexity of the change. For example, if a single node with a single edge is added to a network, one iteration of the algorithm is required which impacts only the added node as it joins the community it is connected to. On the other hand, if a node with many edges joining several communities is added, more processing is required. As an example, node 35 is added to the karate network. This node has multiple edges into both the yellow and green communities. Further, a triad is formed between node 35 and nodes in the green community. Figure 5 shows how the community structure is changed by this addition.

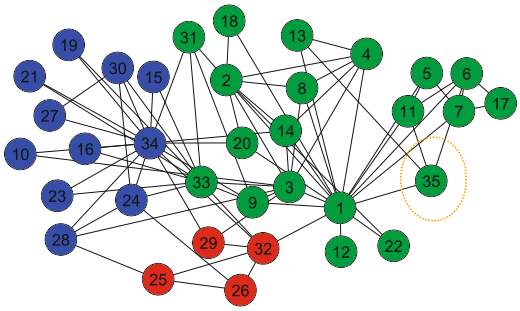


Fig. 5 Impact of adding disruptive node to Zachary's Karate Club Network

The addition of node 35 results in the yellow community being absorbed by the green community. Only four iterations were required for the network to stabilize, and six nodes are impacted. The four iterations reflect a cascading effect as the

nodes most attached to the green community decide to change communities, and the remaining nodes follow. The important characteristic to note is the update to community structure is localized to the affected nodes. No global calculations which span the entire network are required.

5 Conclusions

We have introduced a self-organizing community detection algorithm which reflects the high efficiency and adaptivity of decentralized systems observed in nature. The algorithm is based on localized node entropy calculations which require only knowledge of each node's neighborhood. The entropy calculation reveals community overlap. It also shows the level of "belongingness" or "fuzziness" of nodes in the communities to which they belong. In a demonstration of community partition quality, the algorithm was shown to successfully identify a high resolution partitioning of the Zachary's Karate Club Network. Finally, the algorithm is well suited to manage dynamically changing networks. Network perturbations are localized and the processing required for each update is proportional to the complexity of the update.

References

1. Bastian, M., Heymann, S., Jacomy, M.: Gephi: An open source software for exploring and manipulating networks. In: International AAAI Conference on Weblogs and Social Media (2009)
2. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment (10), P10008+ (2008)
3. Cruz, J.D., Bothorel, C., Poulet, F.: Entropy based community detection in augmented social networks. In: CASoN, pp. 163–168. IEEE (2011)
4. Fortunato, S.: Community detection in graphs. Physics Reports 486, 75–174 (2010)
5. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. PNAS 104, 36 (2007)
6. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. PNAS 99(12), 7821–7826 (2002)
7. Kleinberg, J.: The small-world phenomenon: an algorithm perspective. In: Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, STOC 2000, pp. 163–170. ACM, New York (2000)
8. Ma, X., Gao, L., Yong, X.: Eigenspaces of networks reveal the overlapping and hierarchical community structure more precisely. Journal of Statistical Mechanics: Theory and Experiment (08), P08012 (2010)
9. Mamei, M., Menezes, R., Tolksdorf, R., Zambonelli, F.: Case studies for self-organization in computer science. J. Syst. Archit. 52(8), 443–460 (2006)
10. Newman, M.: Modularity and community structure in networks. PNAS 103(23), 8577–8582 (2006)

11. Ning, H., Xu, W., Chi, Y., Gong, Y., Huang, T.S.: Incremental spectral clustering by efficiently updating the eigen-system. Pattern Recogn. 43(1), 113–127 (2010)
12. Reichardt, J., Bornholdt, S.: Detecting fuzzy community structures in complex networks with a potts model. Physical Review Letters 93(21), 218701 (2004)
13. Shannon, C., Petigara, N., Seshasai, S.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423 (1948)
14. Shi, J., Malik, J.: Normalized cuts and image segmentation. In: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR 1997), IEEE Computer Society, Washington, DC (1997)
15. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of Anthropological Research 33, 452–473 (1977)

Extending Jason with Promises for Concurrent Computation

Alex Muscar

Abstract. Even though the agent-oriented paradigm (AOP) has lost some of its charm in the past couple of years, the agent community is still active and a large variety of real world applications have been developed lately. Ranging from web applications to mobile applications, the paradigm has shown it is a viable choice. From an overview of these applications Jason seems to be the most widely used AOP language. But, while the core foundation of Jason, the Belief-Desire-Intention (BDI) theory, has gotten a lot of attention over the years, the language is still lacking with respect to some practical aspects such as concurrent programming. In this paper we propose an extension to Jason that makes concurrent programming easier with the aid of promises. This extension makes it possible to express concurrent flows in a more natural way. We first present a non-intrusive extension that enables this style of programming and motivate its usefulness. Then we propose a language extension that avoids the inversion of control problem inherent when using promises. We also take into account some of the drawbacks of our proposed approach and investigate some possible solutions.

Keywords: agent-oriented programming, concurrent programming, asynchronous programming, promises.

1 Introduction

Even though the agent-oriented paradigm (AOP) has lost some of its charm in the past couple of years, the agent community has developed real world applications using it, and more specifically the Jason language [2], ranging from web applications [10] to mobile applications [12], thus establishing it is a viable choice. But, while the core foundation of Jason, the Belief-Desire-Intention (BDI) theory, has gotten

Alex Muscar
University of Craiova, Blvd. Decebal, nr. 107, RO-200440, Craiova, Romania
e-mail: amuscar@software.ucv.ro

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 41–50.
springerlink.com

a lot of attention over the years, the language is still lacking with respect to some practical aspects such as concurrent programming. By adding support for concurrent computation in agent languages a whole new area of applications would open up for AOP, from scientific applications like massively parallel simulations to financial applications like *high frequency trading* [13].

Given the ascent of multicore computers and mobile devices, concurrency has become an important issue, and we, as a community, cannot go on ignoring "the elephant in the room". There have been some recent efforts to address this issues such as the one proposed by Ricci et al. in [11], but there is still place for other approaches. An early and interesting example of an agent oriented programming language featuring concurrent computation is Go! [4], which has unfortunately mostly gone unnoticed by the community.

In this paper we propose an extension to Jason that makes concurrent programming easier by using the concept of *promises* introduced by Friedman and Wise in [5]. Later, languages like E[1] and Alice ML[2] adopted the concept and popularized it [1, 8]. The proposed extension makes it possible to express concurrent flows in a more natural way. While this work does not focus on distributed systems, concurrency is inherently present in such scenarios, so tackling this problem, even in the context of single agents or agents running on the same machine, will benefit them as well.

This investigation was prompted (and is part of) a larger research project concerning the development of a dynamic negotiation mechanism and an accompanying framework. We decided to use Jason for implementing an initial prototype, but given the distributed nature of our framework we were soon faced with some of Jason's limitations whose nature we will illustrate in sec. 2.

Before we continue a note on terminology is due. The terms *promise* and *future* are used to refer to constructs used to facilitate concurrent programming. While similar, the terms stand for different mechanisms. Both terms stand for objects that represent the (yet unknown) result of an ongoing computation which is executing concurrently with other computations in the system. The difference is that in order to retrieve the value from a future one has to call a `get` method which will block the calling computation until the value of the future is available. Promises on the other hand, can have *callbacks* attached which will be called when the value of the promise becomes available (i.e. the promise gets *resolved*). We find that promises are more in line with the nature of Jason.

We first present a non-intrusive extension that enables this style of programming and motivate its usefulness. Then we propose a language extension that avoids the transformation of the program into explicit *continuation passing style* (CPS) [15] inherent when using promises. We also take into account some of the drawbacks of our proposed approach and investigate some possible solutions.

This paper is structured as follows: in section 2 we illustrate the problem we are addressing by means of a simple example which we will come back to in later

[1] `http://www.erights.org/`

[2] `http://www.ps.uni-saarland.de/alice/`

sections. In sec. 3 we present a non-intrusive solution based on explicit promises. Then, in sec. 4 we propose an extension of this approach by slightly changing the Jason language and motivate its usefulness over the approach presented in the previous section. Sec. 5 discusses some of the open issues of the proposed approaches and sketches possible solutions. Finally, we conclude in sec. 6, where we present some possible directions for future iterations of this approach.

2 Background

In order to make the following examples easier to understand we will briefly outline the basic concepts that make up a Jason program. A program is made out of three sections: beliefs, rules and plans. *Beliefs* and *rules* are very similar to facts and rules in Prolog with some notational differences: `&` replaces `,` in conjunctions. *Goals* are introduced by the `!` operator (to be precise the `!` introduces *achievement* goals, Jason also has *test* goals introduced by the `?` operator, but they are not relevant for the purpose of this paper). *Plans*, which follow the generic form `triggering_event : context <- body.`, are intended to handle goals. *Triggering events* match goals and message. The only relevant triggering events for this paper are goal addition and message arrival, denoted by atoms with the `+!` and the `+` prefixes. The belief base can be manipulated with the aid of the `+` and `-` operators, in the *body* of the plan. They add, and delete respectively, a belief from the belief base. Since these operations usually come in pairs, a deletion followed by an addition, Jason offers the shortcut operator `-+` for this purpose. One last piece of information needed to understand the examples used in this paper is related to Jason's use of *internal actions*. They allow the extension of the Jason interpreter by using Java. Internal actions are qualified by their package name (like in Java, e.g. `package.internal_action`). Actions pre-defined by the Jason distribution don't have a package name, but they retain the leading period (e.g. `.send`). For further details we direct the interested readers to [3].

With the syntactic details of Jason programs out of the way we are going to describe a simple scenario which we will use as a running example to illustrate the problem we are addressing. As we mentioned in sec. 1 this research is part of a bigger project, and it was prompted by the specific problems that we encountered while developing an initial prototype in Jason. But, since the purpose of this paper is to explore the extension of Jason with promises and because we do not want to obscure the example with the details of the negotiation framework (which is out of the scope of this paper) we will employ a simple scenario which involves a single agent working with two social networks: Facebook and Twitter. Its job is to correlate wall posts and tweets for the user[3]. Since both operations can have considerable

[3] While admittedly synthetic, there is no reason why this example could not be scaled to multiple (eventually distributed) agents. Also, while in this example we use agents as simple reactive entities, this is not a limitation of the proposed approach, but a consequence of our desire to keep the example simple.

delays it would be desirable to i) run both of them concurrently, and ii) not to block the agent while doing so.

A straight forward implementation in Jason would involve two additional agents which act as clients for the social networks. Let us call these agents `facebook_client` and `twitter_client`. Note that for the rest of this paper we assume the presence of two user-define internal actions, `facebook.get_wall_posts` and `twitter.get_tweets`, which allow agents to interface with the social networks. In this setup, the main agent would send a message to each of the client agents to fetch the relevant data. While this obeys both previous requirements, it does so at an added cost for the programmer: manually synchronizing the responses from the clients. Figure 1 shows a possible implementation of this approach.

```
!start.

+!start
  <- .send(facebook_client, tell, get_wall_posts);
     .send(twitter_client, tell, get_tweets).

+!correlate
  : have_wall_posts(Posts) & have_tweets(Tweets)
  <- ...

+wall_posts(Posts)
  <- +have_wall_posts(Posts);
     !correlate.

+tweets(Tweets)
  <- +have_tweets(Tweets);
     !correlate.
```

```
// Facebook client

+get_wall_posts[source(A)]
  <- facebook.get_wall_posts(WallPosts);
     .send(A, tell, WallPosts).

// Twitter client

+get_tweets[source(A)]
  <- twitter.get_tweets(Tweets);
     .send(A, tell, Tweets).
```

Fig. 1 Main agent and clients

Because we need the responses from two concurrently running operations to continue, and because we don't know in which order they we will receive them we use the knowledge base for synchronization: when the response from one client has arrived we store it in the knowledge base and test if the other client has already sent its response. Only when both clients are done can the main agent go on with its computation. In this scheme the synchronization is scattered in three different places: the event handlers for the client responses (`+wall_posts` and `+tweets`), the plan for processing the data (`+!correlate`), and the belief base. Fig. 2 illustrates the interactions between the agents in this approach.

While this technique leads to the desired behavior, it is not optimal for a couple of reasons:

1. It splits the logic of the program over several execution units – in our case the logic really belongs in the main agent, but it is split over the main agent, and the Facebook and Twitter client agents – which makes it hard to have a global view of the program behavior, especially for more complex scenarios, e.g. where the client agents need access to the belief base of the main agent;

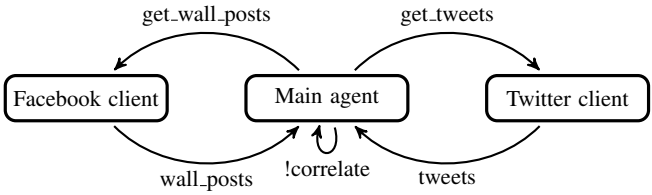


Fig. 2 Interactions between the three agents in the straight forward implementation

2. It uses one client agent for each such request which leads to inefficient and hard to manage programs. While several strategies are possible – starting all the client agents up-front, dynamically creating and destroying them, or managing a pool of such agents – none of them is optimal, the first two because of the cost of running extra agents, and the third because of the extra complexity involved in managing a pool of client agents; and
3. It does not scale: having to manually store the responses of clients as beliefs for many asynchronous requests and synchronizing them by hand is a tedious and error prone task and it leads to mostly duplicate code for handling asynchronous responses (e.g. `+wall_posts`, `+tweets`).

It is clear that if we want a scalable solution we need to look for an alternative approach.

3 Explicit Promises

The solution we propose is to introduce a couple of internal actions for handling concurrently running tasks. By exposing the potential results of the computations as promises this solution allows for a more natural flow.

Our approach is illustrated in fig. 3.

```
!start.

+!start
  <- facebook.get_wall_posts(WallPostsPromise);
     twitter.get_tweets(TweetsPromise);
     async.when(WallPostsPromise, TweetsPromise, correlate).

+!correlate(WallPosts, Tweets)
  <- ...
```

Fig. 3 Non intrusive promises

The first change is to modify `facebook.get_wall_posts` and `twitter.get_tweets` to expose promises and immediately return instead of blocking the plan. The second change is the introduction of the `async.when` internal action which takes any number of promises and an atom representing the name of the plan to trigger when *all* the promises are fulfilled. Note that as the previous actions, the `async.when` combinator does not block the execution of the plan.

Thanks to these changes the implementation of the sample scenario is shorter, but, more importantly, it's conceptually cleaner and it has better code locality, i.e. all the synchronization takes place in the `+!start` plan, instead of being scattered over multiple plans. Furthermore our solution needs no additional agents and is, as such, more scalable.

3.1 Implementation Details

For our implementation we used the utilities in the `java.util.concurrent`[4] package and the Guava libraries[5] which extend the default functionality in `java.util.concurrent` with *listenable futures*. Listenable futures, are futures to which the user can subscribe by providing a callback that gets called when the value of the future is available – i.e. they implement promise semantics.

Our implementation is built around the *thread pool* pattern, represented by an *executor service*. Asynchronous internal actions are scheduled and eventually run by the threads in the thread pool.

As we mentioned before, in order to implement the prototype for our solution we extended the Jason language with an internal action for synchronizing multiple asynchronous operations and a base class for asynchronous internal actions (such as `facebook.get_wall_posts` and `twitter.get_tweets`)– `AsyncInternalAction`. This class subclasses `DefaultInternalAction`, the base class for internal actions in Jason, and overrides its `execute()` method. Instead it defines its own `apply()` method which is where the internal action implementer defines the behavior. When the asynchronous internal action gets executed, the `apply()` method gets wrapped in a `Callable`, and the `ListenableFuture` returned by the executor is stored in static *global future store*. The global store associates a unique identifier to each task. This identifier is returned to the plan via an output variable sent as a parameter to the internal action.

The `async.when` internal action receives a variable number of promise identifiers (as returned by the global future store) and a symbol representing the name of the plan that is going to get triggered once *all* the promises get resolved. When this happens an internal event is constructed with the values of all the promises as parameters.

[4] `http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/package-summary.html`

[5] `http://code.google.com/p/guava-libraries/`

4 Implicit Promises

While the explicit promise approach is a vast improvement over the ad hoc synchronization method that we presented in sec. 2, there is still a problem with it: the callback for the promises still have to be defined as different plans. This forces the developer to program in an explicit continuation passing style[6] with the plan selected for execution by the `when` combinator being the continuation of the promises.

In order to address this issue a slight modification of the Jason language would be necessary. First we would introduce a || operator for concurrent execution. Using it, one could compose the two actions to fetch Facebook posts and tweets mentioned earlier in the following way:

```
facebook.get_wall_posts(WallPosts) || twitter.get_tweets(Tweets)
```

Note that `WallPosts` and `Tweets` are not promises as in the previous examples but output variables for the actions, the promises are implicit. This allows for a more natural style of programming.

In order to synchronize concurrent computations we have two options: either introduce a new operator, say ⇒, or overload the `;` operator. We think the latter is a more natural choice for Jason. Using `;` to stand for either normal, sequential composition, and as an implicit synchronization point for promises our running example could be rewritten as shown in fig. 4.

```
!start.

+!start
  <- facebook.get_wall_posts(WallPosts) ||
     twitter.get_tweets(Tweets);
     !correlate.
```

Fig. 4 Implicit promises

In order to implement this we can employ an approach similar to the one used by the F# language[7] for its *asynchronous computation expressions* [16]. The compiler performs a code rewrite of plans that use the || operator. While this technique suffices in the case of F# thanks to its support for anonymous functions that close over variables in their scope (closures), their lack in Jason means that we have to perform an additional transformation, *lambda lifting* [6]. This transformation identifies the variables that a closure would capture and transforms the closure in a top level function with an extra parameter for each captured variable. Using these techniques, the code in fig. 4 would be rewritten to look like the one in fig. 3 (modulo some identifiers which would have to be generated by the compiler).

[6] Linear control flow is replaced by a scheme where each function call (or predicate/plan) receives an additional functional argument, its "continuation", witch is called instead of normally returning a value.

[7] `http://research.microsoft.com/en-us/um/cambridge/projects/fsharp/`

5 Open Issues

While both approaches presented in the previous sections have a series of advantages when it comes to concurrent computations in Jason they also introduce a series of problems. The main drawback is that the belief base of an agent represents its global shared state. This is not a problem with regular Jason agents because the interpreter serializes access to the belief base, but, by using external mechanisms to execute actions concurrently, e.g. the `java.util.concurrent` package and the Guava library, we circumvent this mechanism. This can lead to the typical problems associated with concurrency, e.g. race conditions and deadlocks.

In the case of explicit promises it is mostly a problem of programmer discipline: asynchronous internal actions should not modify the belief base. Instead, all modifications to the belief base should be performed from the body of a plan. Since only the asynchronous internal actions circumvent the serialization mechanisms of the interpreter this is a safe approach[8].

The case of implicit promises is more problematic since we would like to avoid artificial restrictions and avoid restricting the || operator to asynchronous internal actions. Ideally the programmer should be allowed to use goals or belief base modifications as operands. This is illustrated by the code in fig. 5. Suppose this code is part of an agent in charge of running a coffee machine. When the user sends a request, the agent displays a message to the user and starts making the coffee at the same time.

```
+user_request(make_coffee)
   <- (lcd.display("Preparing coffee"); -+state(making_coffee)) ||
      !make_coffee.
```

Fig. 5 Dangers of global shared state

The key point here is the change in the belief base `-+state(making_coffee)`. This removes any old belief for `state`, and adds a new one with the argument `making_coffee`. If another concurrently running computation were to change the state at the same time, the belief base might be left in an inconsistent state, a classic example of a race condition.

There are a couple of alternatives to address this issue:

- Based on the insight that not all the plans need all the beliefs, we could change the language to introduce local plan beliefs. This is the approach taken by the SimpAL language [11]. While this is an interesting approach for structuring an agent, we feel it would be a disruptive change for the Jason language; and
- *Software Transactional Memory* (STM) is a synchronization mechanism akin to transactions in databases [14]. It is an interesting alternative as it is only an infrastructure change, so it would not surface in the language syntax. But STM has the drawback that the performance of the application may suffer.

[8] But then again, programmers are not the most disciplined of men.

6 Conclusion and Future Work

In this paper we presented an non-intrusive extension for asynchronous computations in Jason using implicit promises and we proposed a language extension that would make the promises implicit, leading to a gain in expressivity.

While somewhat restricted, even the extension featuring explicit extensions proves to be a significant improvement over the straight forward implementation in Jason. Being able to easily compose asynchronous computations offers a big advantage for real world scenarios where agents need to use resource that imply latencies, e.g. web services.

Our next step will be to investigate an implementation of the scenario featuring implicit promises and the related synchronization issues. We find the idea of integrating BDI and STM interesting because it may be the first step toward speculatively parallelized plans. But this a future direction, and for the moment we will focus on the explicit concurrency offered by the || operator.

The next step after implementing the language extension for implicit promises would be to investigate other asynchronous control constructs based on promises. Some interesting work has already been done for other languages like E and Scala[9].

A slight variation on the theme of promises would be to follow the model proposed by some languages like Alice ML and C++11 to separate promises into a listener and a resolver. The listener is the part which can have a callback registered for when the promise is resolved, while a resolver is used to resolve the promise. This dichotomy seems worth while investigating in the context of agents since it enables the decoupling of consuming and resolving promises, having the potential to split promises over agents should that be necessary. This opens up interesting venues like using the *object capability model* [7, 9] for access control in a multi-agent system.

Acknowledgements. This work was supported by the strategic grant POSDRU/CPP107/DMI1.5/S/78421, Project ID 78421 (2010), co-financed by the European Social Fund - Investing in People, within the Sectoral Operational Programme Human Resources Development 2007 - 2013.

References

1. Aspinall, D., Stark, I.: Futures and promises in alice ml (2008), http://www.inf.ed.ac.uk/teaching/courses/apl/2010-2011/examples/aliceml.pdf
2. Bordini, R.H., Hubner, J.F., Vieira, R.: Jason and the golden fleece of agent-oriented programming. In: Bordini, R.H., Dastani, M., Dix, J., Fallah-Seghrouchni, A.E. (eds.) Multi-Agent Programming, Multiagent Systems, Artificial Societies, and Simulated Organizations, vol. 15, pp. 3–37. Springer (2005)
3. Bordini, R.H., Wooldridge, M., Hubner, J.F.: Programming Multi-Agent Systems in AgentSpeak using Jason. Wiley Series in Agent Technology. John Wiley & Sons (2007)

[9] http://asyncobjects.sourceforge.net/asyncscala/index.html

4. Clark, K.L., McCabe, F.G.: Go! - a multi-paradigm programming language for implementing multi-threaded agents. Annals of Mathematics and Artificial Intelligence 41(2-4), 171–206 (2004)
5. Friedman, D., Wise, D.: The Impact of Applicative Programming on Multiprocessing. Technical report (Indiana University, Bloomington. Computer Science Dept.). Indiana University, Computer Science Department (1976)
6. Johnsson, T.: Lambda lifting: transforming programs to recursive equations. In: Proc. of a Conference on Functional Programming Languages and Computer Architecture, pp. 190–203. Springer-Verlag New York, Inc., New York (1985)
7. Miller, M.S.: Robust composition: towards a unified approach to access control and concurrency control. Ph.D. thesis, Baltimore, MD, USA (2006) AA13245526
8. Miller, M.S., Tribble, E.D., Shapiro, J.S.: Concurrency Among Strangers: Programming in E as Plan Coordination. In: De Nicola, R., Sangiorgi, D. (eds.) TGC 2005. LNCS, vol. 3705, pp. 195–229. Springer, Heidelberg (2005)
9. Miller, M.S., Yee, K.P., Shapiro, J.: Capability Myths Demolished. Tech. rep., Systems Research Laboratory, Johns Hopkins University (2003)
10. Minotti, M., Piancastelli, G., Ricci, A.: Agent-Oriented Programming for Client-Side Concurrent Web 2.0 Applications. In: Cordeiro, J., Filipe, J. (eds.) WEBIST 2009. LNBIP, vol. 45, pp. 17–29. Springer, Heidelberg (2010)
11. Ricci, A., Santi, A.: Designing a general-purpose programming language based on agent-oriented abstractions: the simpal project. In: Proceedings of the Compilation of the Co-located Workshops on DSM 2011, TMC 2011, AGERE! 2011, AOOPES 2011, NEAT 2011, VMIL 2011, SPLASH 2011 Workshops, vol. 38, pp. 159–170. ACM, New York (2011), doi:10.1145/2095050.2095078
12. Santi, A., Guidi, M., Ricci, A.: JaCa-Android: An Agent-Based Platform for Building Smart Mobile Applications. In: Dastani, M., El Fallah Seghrouchni, A., Hübner, J., Leite, J. (eds.) LADS 2010. LNCS, vol. 6822, pp. 95–114. Springer, Heidelberg (2011)
13. International Organization of Securities Commissions, T.C.: Regulatory issues raised by the impact of technological changes on market integrity and efficiency. Tech. rep., International Organization of Securities Commissions (2011)
14. Shavit, N., Touitou, D.: Software transactional memory. Distributed Computing 10(2), 99–116 (1997)
15. Sussman, G.J., Steele Jr., G.L.: Scheme: an interpreter for extended lambda calculus. MIT AI Memo, vol. 349. Massachusetts Institute of Technology, Cambridge (1975)
16. Syme, D., Petricek, T., Lomov, D.: The F# Asynchronous Programming Model. In: Rocha, R., Launchbury, J. (eds.) PADL 2011. LNCS, vol. 6539, pp. 175–189. Springer, Heidelberg (2011)

Dynamic Case-Based Reasoning Based on the Multi-Agent Systems: Individualized Follow-Up of Learners in Distance Learning

Abdelhamid Zouhair, El Mokhtar En-Naimi, Benaissa Amami, Hadhoum Boukachour, Patrick Person, and Cyrille Bertelle

Abstract. In a Computing Environment for Human Learning (CEHL), there is still the problem of knowing how to ensure an individualized and continuous learner's follow-up during learning process, indeed among the numerous methods proposed, very few systems concentrate on a real time learner's followup. Our work in this field develops the design and implementation of a Multi-Agent Systems Based on Dynamic Case Based Reasoning which can initiate learning and provide an individualized follow-up of learner. When interacting with the platform, every learner leaves his/her traces in the machine. These traces are stored in a basis under the form of scenarios which enrich collective past experience. The system monitors, compares and analyses these traces to keep a constant intelligent watch and therefore detect difficulties hindering progress and avoid possible dropping out. The system can support any learning subject. To help and guide the learner, the system is equipped with combined virtual and human tutors.

Keywords: Distance Learning, Dynamic Case-Based Reasoning, Intelligent Tutor, Multi-Agent Systems, Scenarios, Traces.

1 Introduction

The CEHL is a computer tool which offers learners another medium of learning. Indeed it allows learner to break free from the constraints of time and place of training. They are due to the learner's availability. In addition, the instructor is not physically present and training usually happens asynchronously. However, most E-

Abdelhamid Zouhair · Hadhoum Boukachour · Patrick Person · Cyrille Bertelle
LITIS, The University of Le Havre, France
e-mail: abdelhamid.zouhair@litislab.fr

El Mokhtar En-Naimi · Benaissa Amami
LIST Laboratory, The FST of Tangier, Morocco
e-mail: ennaimi@gmail.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 51–57.
springerlink.com

learning platforms allow the transfer of knowledge in digital format, without integrating the latest teaching approach in the field of education (e. g. constructivism, [13], ...). Consequently, in most cases distance learning systems degenerate into tools for downloading courses in different formats (pdf, word ...). These platforms also cause significant overload and cognitive disorientation for learners. Today, it is therefore necessary to design a CEHL that provides individualized follow-up to meet the pace and process of learning for the learner, who thus becomes the pilot of training. Our contribution in this field is to design and implement a computer system (i. e. intelligent tutor) able to initiate the learning and provide an individualized monitoring of the learner.

Solving these problems involves first, to understand the behavior of the learner, or group of learners, who use CEHL to identify the causes of problems or difficulties which a learner can encounter. This can be accomplished while leaning on the traces of interactions of the learner with the CEHL, which include history, chronology of interactions and productions left by the learner during his/her learning process. This will allow us the reconstruction of perception elements of the activity performed by the learner. According to Marty and Mille [10] the digital traces of interactions represent a major resource customization CEHL.

We propose a system able to represent, follow and analyze the evolution of a learning situation through the exploitation and the treatment of the traces left by the learner during his/her learning on the platform. This system is based, firstly on the traces to feed the system and secondly on the reconciliation between the course of the learner (traces in progress) and past courses (or past traces). The past traces are stored in the form of scenarios in a database called ”base of scenario”. Recently, several research works have been focused on the dynamic case based reasoning in order to push the limits of case based reasoning system dealing with situations static, reactive and responsive to users. All these works are based on the observation that the current tools are limited in capabilities, and are not capable of evolving to fit the non-anticipated or emerging needs. For example, few CBR systems are able to change over time the way of representing a case [12]. We propose a system, which analyzes the traces of learners in a continuous way, in order to ensure an automatic and a continuous monitoring of the learner. Our work in this field develops the design and implementation of a Dynamic Case Based Reasoning founded on the Multi-Agent Systems (MAS).

The rest of this paper is organized as follows: In the second section, we give a general introduction of E-earning and intelligent tutoring. The third section is devoted to the presentation of the design and implementation of our approach. So we will introduce the general architecture of the system. In section four, we will describe the approach of Multi-Agent Case Based Reasoning. In the following section, we will propose the description of our approach Multi-Agent Dynamic Case Based Reasoning. Finally, we will give the conclusion and our perspectives.

2 Intelligent Tutor and Distance Learning

Intelligent Tutoring Systems (ITS) are computer systems designed to assist and facilitate the task of learning for the learner. They have expertise in so far as they know the subject matter taught (domain knowledge), how to teach (pedagogical knowledge) and also how to acquire information on the learner (learner representative).

There is much research concerned with the design and implementation of computer systems to assist a learner in learning. There are, for example, tutors or teaching agents who accompany learners by proposing remedial activities [5]. There are also the agents of support to the group collaboration in the learning [3] encouraging, the learners participation and facilitating discussion between them. Other solutions are based on agents that incorporate and seek to make cooperation among various Intelligent Tutoring Systems [2]. The Baghera platform [18], which is a ”distance” CEHL exploits the concepts and methods of Multi-Agent approach. Baghera assists learners in their work solving exercise in geometry. These tools of distance learning do not allow an individualized, continuous and real-time learners follow-up. They adopt a traditional pedagogical approach (behaviorist) instead of integrating the latest teaching approaches (constructivism [13], [17]). Finally, given the large number of learners who leave their training, the adaptation of learning according to the learners profile has become indispensable today. Our contribution consists in proposing an adaptive system to ensure an automatic and a continuous monitoring of the learner. This monitoring is based on cases (dropping out, difficulties) past and similar.

3 General Architecture of our System

One of the main objectives of the individualized monitoring of the learner is to envisage, to anticipate and to reduce the number of dropping out, which makes us seek a flexible and adaptive solution [4]. The complexity of the situations to be treated leads us to choose an approach based on a MAS, able to cooperate and coordinate their actions to provide a pedagogical adaptation for the learners profile. We reconcile the problems of the analysis of the traces left by the learners activity in CEHL, and the decision support systems, able to represent, follow in real-time and analyze the evolution of a dynamic situation. Such a system must represent the current situation, take into account the dynamic change of the current situation, predict the possible evolution of this situation, and react depending on the particular situations and the learners profiles. This can be done by using past situations which consequences are known. It is then a question of reasoning by analogy. This type of reasoning can allow solving new problems, using already solved problems available in memory. The system we propose, allows to analyze the learners course (trace) in order to anticipate a possible dropping-out. The learning activities past traces will be the source of knowledge for the learning adaptation process, they are stored in a database called ”base of scenarios”. Each scenario contains all determining aspects in its development, i.e, the facts that have played an effective role in the way the events proceeded. The analysis of the current situation must be continuous and

dynamic. Indeed, the target case is a plot that evolves, therefore the system must take this incremental evolution into account.

The general architecture of our system consists of the three following components (Fig. 1):

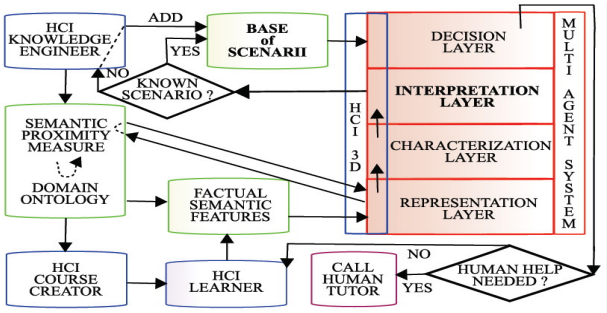


Fig. 1 General architecture of our intelligent tutor system

- The graphical interfaces for learners (who are the users for whom the system is developed), for course designers (who must structure the teaching contents) and finally the developers (Human and Computer Interface "HCI" knowledge engineer for the knowledge module, and a tree Dimension Human and Computer Interface "3D HCI" for the behavior of the Multi-Agent Systems);
- The Knowledge module containing: Base of Scenarios, Factual Semantic Features, Semantic Proximity Measure and Domain Ontology;
- The hierarchical MAS with four layers.

4 Multi-Agent Case Based Reasoning

Case-Based Reasoning (CBR) is an artificial intelligence methodology which aims at solving new problems based on past experience or the solutions of similar previous problems in the available memory [7]. The solved problems are called source cases and are stored in a database (called a case-base or base of scenarios). The problem to be solved is stored as a new case and is called target case. The systems based on the case-based reasoning can be classified into two categories of applications [9]: Applications dealing with situations static (i. g. CHIEF [6] and Creek [1]) and Applications with dynamic situations (i. g. REBECAS [9]), for more details on the subject, the reader may refer to [9].

The Multi-Agent Systems based on case based reasoning are used in many applications areas. we can distinguish two types of applications:

- The Multi-Agent Systems in which each agent uses the case based reasoning internally to their own needs (level agent case based reasoning). For this systems, each agent is able to find similar cases to the target case in their own case base,

also able to accomplish the other steps of CBR cycle (i. g. the system AMAL [15] for Multi-Agent arguments, CCBR for personalized route planning [11], and MCBR [8] for distributed systems).

- The Multi-Agent Systems whose approach is a case based reasoning (level Multi-Agent Case Based Reasoning). For this applications, the Multi-Agent Case Based Reasoning System distribute the some/all steps of the CBR cycle (Representation, Retrieve, Reuse, Revise, Retain) among several agents. This type of approach might be better than the first. Indeed, the individual agents experience may be limited, therefore their knowledge and predictions too, thus the agents can benefit from the other agents capabilities, cooperate with each other for better prediction of the situation (i. g. the PROCLAIM [16] in argumentation field, and CBRTEAM [14] in a parametric design task).

5 Multi-Agent Dynamic Case Based Reasoning

Our problem is similar to the CBR for dynamic situations. Indeed, the traces left by the learner during the learning session evolve dynamically over time; the case-based reasoning must take into account this evolution in an incremental way. In other words, we do not consider each evolution of the traces as a new target. The CBR which we propose offer important features: (1) It is dynamic. Indeed, we must continually acquire new knowledge to better reproduce human behavior in each situation; And (2) It is incremental, this is its major feature because the trace evolves in a dynamic way for the same target case. The main benefits of our approach are the distributed capabilities of the Multi-Agent Systems and the self-adaption ability to the changes that occur in each situation.

Each action of the learner is represented by a data structure called semantic features that are supported by factual agents. The course of the learner is well represented by a set of trace agents [4]. Therefore, the various actions of the learner (learner traces) can be represented as a collection of semantic features. These will feed the representation layer (Layer 1). The role of this layer is to be both, a picture of the current situation being analyzed and to represent the dynamics of its evolutions over time. The goal of the characterization layer (Layer 2) is to provide a synthetic vision of the organization of agents of the representation layer by classifying them in several subsets according to their activity degrees. A part of the target case in the dynamic and incremental case-based reasoning is developed by this layer. The interpretation, or prediction, layer (Layer 3) will associate the agents characterization subsets layer with a scenario. The interpretation agents also allow to update the system knowledge by the learning of new cases. In fact, they store and manage new scenarios [4]. The decision layer (Layer 4) selects similar scenarios in the base of scenarios and chooses one to propose to the learner. For each particular situation, the decision agents can react differently depending on the learners profile concerned, for example, deciding to initiate a communication session with a learners experiencing difficulties. The human tutor is needed if the system detects a learning situation requiring his/her intervention.

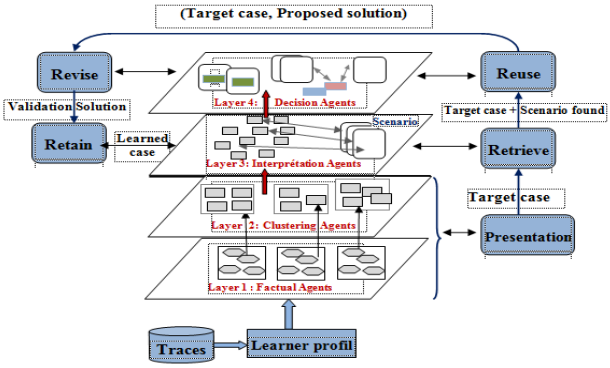


Fig. 2 Dynamic CBR cycle in our approach

The CEHL personalization is primarily depending on the ability to produce relevant and exploitable traces of the learners activity (for more details concerning the learner's traces, the reader may refer to [19]). These traces allow us to describe and to document the learners activity. In the current uses of the traces for the CEHL, collected situations are contrasted: from "we take what we have in well specified formats, what is called the logs" to "we scrupulously instruments the environment to recover the observed controlled and useful for different actors (learner and tutor)". The first step consists of modeling the raw data contained in the log file. It is necessary to be able to collect files of traces containing at least, the following elements: time for the start date of the action, codes action which consists in codifying the learners actions and learner concerned.

6 Conclusion and Future Work

Our system allows connecting and comparing the scenario found (current situation) to past scenarios that are stored in a database. The continuous analysis of information coming from the environment (learners traces) makes it possible to suggest to various actors (learners and tutor) possible evolutions of the current situation. The Multi-Agent architecture that we propose is based on four layers of agents with a pyramidal relation. We have presented systems based on Dynamic Case Based Reasoning and we have also clarified that the CBR-based applications can be classified according to the study area: CBR for static situations and CBR for dynamic situations. In our situation, we have used a dynamic case based reasoning with important features. Indeed, the current situation (target case) is a trace that evolves; the case based reasoning must take into account this evolution incrementally. In other words, it shouldnt consider each evolution of the trace as a new target case. Our future work consists in realizing a comparative study between our system and other tools. In addition, by giving a comparison of different existing similarity measures between sequences, We will propose our new similarity measure, such as: The Inverse LCSS (ILCSS).

References

1. Aamodt, A.: Knowledge-Intensive Case-Based Reasoning in CREEK. In: Funk, P., González Calero, P.A. (eds.) ECCBR 2004. LNCS (LNAI), vol. 3155, pp. 1–15. Springer, Heidelberg (2004)
2. Brusilovski, P.: Distributed Intelligent Tutoring on the Web. In: 8th World Conference of Artificial Intelligence in Education, pp. 482–489. IOS Press (1997)
3. Constantino, G., Suthers, D., Icaza, J.-I.: Designing and Evaluating a Collaboration Coach: Knowledge and Reasoning. In: Proceedings of the AI-ED. IOS Press (2001)
4. Ennaji, M., Boukachour, H., Grav, P.: Une architecture Multi-Agent pour la pdagogie de la formation distance. In: MOSIM 2006, Rabat, Maroc (2006)
5. Frasson, C., Martin, L., Gouarderes, G., Ameur, E.: LANCA: A Distance Learning Architecture Based on Networked Cognitive Agents. In: Goettl, B.P., Halff, H.M., Redfield, C.L., Shute, V.J. (eds.) ITS 1998. LNCS, vol. 1452, pp. 594–603. Springer, Heidelberg (1998)
6. Hammond, K.J.: CHEF: a model of case-based planning. In: Proc. of AAAI 1986, pp. 267–271. Morgan Kaufmann (1986)
7. Kolodner, J.: Case-Based Reasoning. Morgan Kaufmann, San Francisco (1993)
8. Leake, D.B., Sooriamurthi, R.: When Two Case Bases Are Better than One: Exploiting Multiple Case Bases. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 321–335. Springer, Heidelberg (2001)
9. Loriette-Rougegrez, S.: Raisonnement partir de cas pour des volutions spatiotemporelles deprocessus, revue internationale de gomatique 8(1-2) (1998)
10. Marty, J.-C., Mille, A.: Analyse de traces et personnalisations des environnements informatiques pour lapprentissage humain. Edition Lavoisier (2009)
11. McGinty, L., Smyth, B.: Collaborative Case-Based Reasoning: Applications in Personalised Route Planning. In: Aha, D.W., Watson, I. (eds.) ICCBR 2001. LNCS (LNAI), vol. 2080, pp. 362–376. Springer, Heidelberg (2001)
12. Mille, A.: From case-based reasoning to traces-based reasoning. Annual Reviews in Control 30(2), 223–232 (2006) ISSN 1367-5788
13. Piaget, J.: Psychologie et pdagogie. Denol-Gonthier, Paris (1969)
14. Prassad, M., Lesser, V., Lander, S.: Retrieval and reasoning in distributed case bases. Technical report, UMass Computer Science Department (1995)
15. Ontañón, S., Plaza, E.: Learning and Joint Deliberation through Argumentation in Multi-Agent Systems. In: AAMAS 2007, Honolulu, HI, USA (2007)
16. Tolchinsky, P., Modgil, S., Cortes, U., Sanchez-marre, M.: Cbr and argument schemes for collaborative decision making. In: COMMA 2006, vol. 144, p. 7182 (2006)
17. Vygotski, L.-S.: Mind in society: the development of higher psychological processes. Harvard University Press, Cambridge (1978)
18. Webber, C., Pesty, S.: Emergence de diagnostic par formation de coalitions-Application au diagnostic des conceptions d'un apprenant. Journes Francophones Pour l'I. A. D. et les SMA (2002)
19. Zouhair, A., En-Naimi, E.M., Amami, B., Boukachour, H., Person, P., Bertelle, C.: Individualized Follow-up of Learners based on Multi-Agent Case-Based Reasoning in Distance Learning. The International Journal IJICT 4(4) (November 2011)

Distributed Paraconsistent Belief Fusion*

Barbara Dunin-Kęplicz and Andrzej Szałas

Abstract. The current paper is devoted to belief fusion when information sources may deliver incomplete and inconsistent information. In such cases paraconsistent and commonsense reasoning techniques can be used to complete missing knowledge and disambiguate inconsistencies. We propose a novel, realistic model of distributed belief fusion and an implementation framework guaranteeing its tractability.

1 Distributed Beliefs

In contemporary intelligent distributed systems, like multiagent systems, we typically deal with many heterogenous information sources. They independently deliver information (e.g. percepts), expressed in terms of beliefs, on various aspects of a recent situation. Depending on the context and the goal of the reasoning process, different beliefs need to be fused in order to achieve more holistic judgement of the situation. Apparently, this information fusion may be realized in various ways. In this paper we will take a closer look at this formal process.

Distributed information sources naturally introduce four truth values: true (false) indicating that only truth (falsity) of a proposition is claimed, inconsistent indicating that both truth and falsity is claimed and unknown indicating that nothing about truth/falsity is claimed [3, 10, 16, 17]. These truth values are further denoted

Barbara Dunin-Kęplicz
Institute of Informatics, University of Warsaw, Banacha 2, Warsaw, Poland
e-mail: keplicz@mimuw.edu.pl
and
Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

Andrzej Szałas
Department of Computer and Information Science, Linköping University, Linköping, Sweden
and
Institute of Informatics, University of Warsaw, Banacha 2, Warsaw, Poland
e-mail: andrzej.szalas@mimuw.edu.pl

* Supported by the Polish National Science Centre grant 2011/01/B/ST6/02769.

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 59–69.
springerlink.com

by $\mathbf{t}$, $\mathbf{f}$, $\mathbf{i}$, $\mathbf{u}$, respectively. However, in real-world distributed problem solving, lack of knowledge and inconsistent beliefs are to be resolved at some point. More precisely, in the case of lacking knowledge, we need to complete missing information at the objective level, while in the case of inconsistencies we do this at a meta-level, for example, by verifying which information sources deliver false information. This knowledge can be then used for a better setup or calibration of sensors and other data sources, as well as diagnostic systems detecting malfunctioning devices. When this is impossible, especially in time critical systems, commonsense reasoning methods can be of help as they generally characterize typical situations [5, 15, 18]. Among these methods (local) closed world assumption, default reasoning, autoepistemic reasoning and defeasible reasoning are of primary importance. Unfortunately, a characteristic feature of these approaches is their high complexity [6, 7, 11, 14].

The contribution of this paper depends on achieving:

- a practical model of distributed belief fusion,
- implementation framework of distributed belief fusion via epistemic profiles,
- tractability of the method.

To model similar phenomena, a commonly used logic is the four-valued logic proposed in [3]. However, as discussed, e.g., in [9, 23], this approach is problematic as it often provides results deviating from intuitions. Importantly, we apply reasoning over databases rather than over general theories. Such an approach reflects the reality of intelligent systems and significantly reduces the complexity of reasoning, typically from at least exponential to deterministic polynomial time, no matter what type of reasoning is used. This substantial complexity gain is achieved by using 4QL, a query language [16, 17] which enjoys tractable query computation and captures all tractable queries. It provides simple, yet powerful constructs for expressing nonmonotonic rules [16]. Moreover, 4QL has a modular structure: its modules can naturally be distributed among agents. Therefore, 4QL is a natural implementation tool creating a space for a diversity of applications.[2] In fact, our approach is strongly influenced by ideas underlying 4QL.

The paper is structured in the following manner. First, in Section 2, we recall belief structures and epistemic profiles. Next, in Section 3 we proceed with a motivating example. Section 4 contains a brief introduction to 4QL. In Section 5 we discuss the proposed solution. Finally, Section 6 concludes the paper.

2 Belief Structures and Epistemic Profiles

First, we need to recall the four-valued logic we apply. We consider two orderings: *truth ordering* $\mathbf{f} < \mathbf{u} < \mathbf{i} < \mathbf{t}$ and *knowledge ordering* $\mathbf{u} = \mathbf{f} < \mathbf{t} < \mathbf{i}$.[3] Conjunction and disjunction is defined by $p \wedge q \stackrel{\text{def}}{=} \min\{p,q\}$ and $p \vee q \stackrel{\text{def}}{=} \max\{p,q\}$, where min and max are minimum and maximum w.r.t. truth ordering. Implication is defined by $p \rightarrow q \stackrel{\text{def}}{=} p \leq q$, where $\leq$ reflects knowledge ordering.

[2] An interpreter of 4QL, Inter4QL, is available via 4ql.org.

[3] This ordering with the identification of $\mathbf{u}$ and $\mathbf{f}$ is introduced and motivated in [22].

If L is a set of literals then the value of an atomic ground literal p in L is:

$$\begin{cases} \mathfrak{t} \text{ when } p \in L \text{ and } \neg p \notin L, \\ \mathfrak{f} \text{ when } \neg p \in L \text{ and } p \notin L, \\ \mathfrak{i} \text{ when } p \in L \text{ and } \neg p \in L, \\ \mathfrak{u} \text{ when } p \notin L \text{ and } \neg p \notin L. \end{cases}$$

This definition can be extended to cover all formulas in the standard way. For more details and motivations see [16, 17, 22, 23].

Let us now recall the definition of belief structures and epistemic profiles, introduced recently in [10]. By $\textsc{Fin}(S)$ we understand the set of all finite subsets of set S.

Definition 1. Let $\mathbb{C} \stackrel{\text{def}}{=} \textsc{Fin}(\mathscr{G}(Const))$ be the set of all finite sets of ground literals with constants in *Const*. Then:

- by a *constituent* we understand any set $C \in \mathbb{C}$;
- by an *epistemic profile* we understand any function $\mathscr{E} : \textsc{Fin}(\mathbb{C}) \longrightarrow \mathbb{C}$;
- by a *belief structure over an epistemic profile* $\mathscr{E}$ we mean $\mathscr{B}^{\mathscr{E}} = \langle \mathscr{C}, F \rangle$, where:
  - $\mathscr{C} \subseteq \mathbb{C}$ is a nonempty set of constituents;
  - $F \stackrel{\text{def}}{=} \mathscr{E}(\mathscr{C})$ is the *consequent* of $\mathscr{B}^{\mathscr{E}}$. ◁

Observe that a crucial concept is that of an epistemic profile. It serves as an abstraction to belief formation processes covering possible methods of reaching final beliefs on the basis of the initial ones. In [10] it has also been indicated that such structures can be implemented in deterministic polynomial time using 4QL [16, 17]. In the current paper we elaborate on the concept of epistemic profiles and show how can one actually implement them.

The main idea is illustrated in Figure 1. Namely, we propose to implement epistemic profiles via an intermediate layer consisting of *derivatives*, where each derivative is a finite set of ground literals. Intuitively, derivatives represent intermediate belief fusion results or, in other words, intermediate views on the situation in question. Importantly, such a structure allows us to implement belief fusion in a highly distributed manner.

3 A Motivating Example

Consider an agent equipped with a sensor platform for detecting air pollution and two different sensors for measuring the noise level. The agent has also some information about the environment, including streets in the neighborhood, etc. The task is to decide whether conditions in the tested position are healthy.

It is natural to consider, among others, three constituents:

- C_p gathering beliefs about air pollution at given places, in terms of $P(x,y)$ indicating the pollution level y at place x, where $y \in \{low, moderate, high\}$;

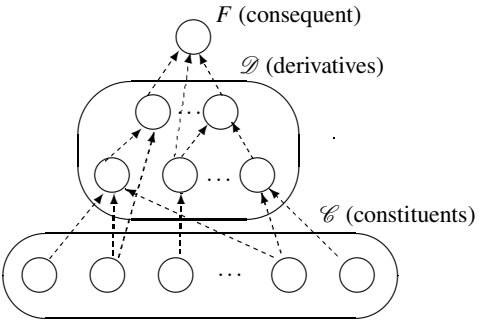


Fig. 1 Implementation framework for belief structures and epistemic profiles. Arrows indicate belief fusion processes.

- C_n gathering beliefs about noise level at given places, in terms of $N_i(x,y)$ indicating the noise level y at place x, as measured by a sensor $i \in \{1,2\}$, where $y \in \{low, moderate, high\}$;
- C_e gathering information about the environment in terms of $Cl(x,y)$ indicating that place x is close to a place with characteristics y, where $y \in \{pollutive, noisy, neutral\}$.

For example, we may have:

$$C_p = \{P(a,low)\},\ \ C_n = \{N_1(a,high)\},$$
$$C_e = \{\neg Cl(a,noisy), Cl(a,neutral), Cl(a,pollutive)\}.$$

Note that we have no information from the second noise sensor (no literal $N_2()$ is given) and somehow inconsistent information as to the pollution level (C_p indicates low level, but according to C_e the agent is close to a pollutive location). Also there is an implicit disagreement between $N_1(a,high)$ appearing in C_e and $\neg Cl(a,noisy)$ appearing in C_e, which may be caused by a defective information source.

One can consider two derivatives:

- D_p – for deciding the pollution level;
- D_n – for deciding the noise level.

Such derivatives should result from reasoning patterns defined by the corresponding epistemic profile of the considered agent. For example, these derivatives may be:

$$D_p = \{P(a,moderate)\},\ \ D_n = \{N(a,high)\}.$$

Based on the contents of D_p and D_n, the agent has to decide whether the situation is healthy or not (and include it in its set of consequents). For example, the agent may accept $F = \{\neg S(a,healthy), S(a,healthy)\}$ as its consequent, i.e., it may have inconsistent beliefs about the issue whether the situation at place a is healthy.

4 Implementation Tool: 4QL

There are several languages designed for programming BDI agents (for a survey see, e.g., [19]). However, none of these approaches directly addresses belief formation, in particular nonmonotonic/defeasible reasoning techniques. Our choice is therefore 4QL, a DATALOG$^{\neg\neg}$-like query language. It supports a modular and layered architecture, and provides a tractable framework for many forms of rule-based reasoning both monotonic and nonmonotonic. As the underpinning principle, openness of the world is assumed, which may lead to the lack of knowledge. Negation in rule heads, expressing negative conclusions, may lead to inconsistencies. As indicated in [16], to reduce the unknown/inconsistent zones, *modules* and *external literals* provide means for:

- the application-specific disambiguation of inconsistent information;
- the use of (Local) Closed World Assumption;
- the implementation of various forms of nonmonotonic and defeasible reasoning.

To express nonmonotonic/defeasible rules we apply modules as well as external literals, originally introduced in [16]. Importantly, different modules can be distributed among different agents participating in the reasoning process.

In the sequel *Mod* denotes the set of *module names*.

Definition 2. An *external literal* is an expression of one of the forms:

$$M.R, -M.R, M.R \text{ IN } T, -M.R \text{ IN } T, \tag{1}$$

where $M \in Mod$ is a module name, R is a positive literal, '$-$' stands for negation and $T \subseteq \{\mathbf{f}, \mathbf{u}, \mathbf{i}, \mathbf{t}\}$. For literals (1), module M is called the *reference module*. ◁

The intended meaning of "$M.R$ IN T" is that the truth value of $M.R$ is in the set T. External literals allow one to access values of literals in other modules. If R is not defined in the module M then the value of $M.R$ is assumed to be $\mathbf{u}$.

Assume a strict tree-like order $\prec$ on *Mod* dividing modules into layers. An external literal with reference module M_1 may appear in rule bodies of a module M_2, provided that $M_1 \prec M_2$.[4]

Definition 3. By a *rule* we mean any expression of the form:

$$\ell \text{ :- } b_{11}, \ldots, b_{1i_1} \mid \ldots \mid b_{m1}, \ldots, b_{mi_m}. \tag{2}$$

where ℓ is a literal, $b_{11}, \ldots, b_{1i_1}, \ldots, b_{m1}, \ldots, b_{mi_m}$ are literals or extended literals, and '$,$' and '$|$' abbreviate conjunction and disjunction, respectively.

Literal ℓ is called the *head* of the rule and the expression at the righthand side of :– in (2) is called the *body* of the rule. ◁

[4] Observe that layers generalize the concept of stratification of DATALOG$^{\neg}$ queries [17] (for definition of stratification see, e.g., [1]).

Table 1 Modules corresponding to constituents considered in Section 3

module Cp:	*module Cn:*	*module Ce:*
domains:	*domains:*	*domains:*
literal level. literal place.	*literal level. literal place.*	*literal type. literal place.*
relations:	*relations:*	*relations:*
P(place, level).	*N1(place, level).*	*Cl(place, type).*
	N2(place, level).	
facts:	*facts:*	*facts:*
P(a, low).	*N1(a, high).*	*– Cl(a, noisy).*
end.	*end.*	*Cl(a, neutral).*
		Cl(a, pollutive).
		end.

Rules of the form (2) are understood as implications:

$$((b_{11} \wedge \ldots \wedge b_{1i_1}) \vee \ldots \vee (b_{m1} \wedge \ldots \wedge b_{mi_m})) \rightarrow \ell,$$

where it is assumed that the empty body takes the value $\mathbf{t}$ in any set of literals.

Definition 4. Let a set of constants, *Const*, be given. A set of ground literals L with constants in *Const* is a *model of a set of rules* S iff each ground instance of each rule of S (understood as implication) obtains the value $\mathbf{t}$ in L. ◁

The semantics of 4QL is defined via well-supported models generalizing the idea presented in [13]. Intuitively, a model is *well-supported* if all derived literals are supported by a reasoning grounded in facts. It appears that for any set of rules there is a unique well-supported model and it can be computed in polynomial time. For details see [16, 17].

Consider now the scenario outlined in Section 3. Exemplary modules corresponding to constituents, derivatives and consequents are shown in Tables 1–3, respectively.[5]

It is important to note that well-supported models are sets of literals. Thus relational or deductive databases technology can be used to query them (see also Section 5.1).

In fact, four logical values, external literals and modular architecture distinguish 4QL from many other approaches (for a survey see, e.g., [2]). 4QL modules are structured in layers. While the lowest layer represents monotonic reasoning, higher ones allow the user to provide rules for disambiguating inconsistencies and completing lacking information. This is achieved by explicitly referring to logical values via external literals.

[5] We use the Inter4QL self-explanatory syntax – for details see 4ql.org.

Table 2 Modules corresponding to derivatives considered in Section 3

```
module Dp:
  domains:
    literal level.   literal place.
  relations: P(place, level).
  rules:
    P(X, moderate) :– Cp.P(X, low) IN {TRUE, INCONS},
                      Ce.Cl(X, pollutive) IN {TRUE, UNKNOWN}.
    ...
end.

module Dn:
  domains:
    literal level.   literal place.
  relations: N(place, level).
  rules:
    N(X,Y) :– Cn.N1(X,Y), Cn.N2(X,Y) IN {TRUE, UNKNOWN} |
              Cn.N1(X,Y) IN {TRUE, UNKNOWN}, Cn.N2(X,Y).
    ...
end.
```

Table 3 Module corresponding to consequents considered in Section 3

```
module F:
  domains: literal characteristics.   literal place.
  relations: S(place, characteristics).
  rules:
    – S(X, healthy) :– Dn.N1(X, high), Dn.N2(X, high).
    S(X, healthy)   :– Cp.P(X, moderate),
                       Cn.N1(X, low) IN {TRUE, UNKNOWN}.
    ...
end.
```

5 Belief Fusion

5.1 Querying Belief Structures

In order to find out what are actual beliefs of agents, groups of agents, etc., the mechanism based on querying belief structures is applicable. Namely, belief structures, when implemented in 4QL, can be considered as sets of ground literals generated by facts and rules. Therefore, one can tractably query belief structures using such query languages as first-order queries, fixpoint queries or 4QL queries, like:

$$\mathrm{Bel}(\exists X(S(X,healthy))) \text{ – is there a healthy place?}$$
$$\mathrm{Bel}(\forall X(S(X,healthy))) \text{ – are all places healthy?}$$

Belief fusion requires gathering beliefs of different agents, e.g.:

$$\begin{array}{l}\mathrm{Bel}_1(\exists X(S(X,healthy))) \wedge \mathrm{Bel}_2(\exists X(S(X,healthy))),\\ \exists X(\mathrm{Bel}_1(S(X,healthy)) \wedge \mathrm{Bel}_2(S(X,healthy))),\\ \mathrm{Bel}_1(\forall X(S(X,healthy))) \vee \mathrm{Bel}_2(\forall X(S(X,healthy))).\end{array}$$

If $Ag_1.S, Ag_2.S$ refer to relation S included in the set of consequents of Ag_1's and Ag_2's belief structures then the above queries can be represented by:

$$\begin{array}{l}\exists X(Ag_1.S(X,healthy)) \wedge \exists X(Ag_2.S(X,healthy)),\\ \exists X(Ag_1.S(X,healthy) \wedge Ag_2.S(X,healthy)),\\ \forall X(Ag_1.S(X,healthy)) \vee \forall X(Ag_2.S(X,healthy)),\end{array}$$

where $Ag_i.S$ indicates that the value of S is taken from consequents of agent Ag_i.

5.2 *Nonmonotonic Reasoning*

Nonmonotonicity is typically caused by attempts to fill gaps in missing beliefs by:

- efficient representation of negative information (Closed World Assumption);
- drawing rational conclusions from non-conclusive information (circumscription, default logics), or from the lack of knowledge (autoepistemic reasoning);
- resolving inconsistencies (defeasible reasoning).

Reasoning in traditional nonmonotonic logics is not tractable [6, 7, 11, 14]. However, their rule-based counterparts appear to be tractable [16]. As 4QL captures all queries expressible in deterministic polynomial time [17], it allows one to encode tractable fragments of various forms of nonmonotonic reasoning.

5.2.1 Local Closed World Assumption

One of the most frequent nonmonotonic techniques are variants of (Local) Closed World Assumption [12]. Intuitively, one often wants to contextually close a chosen part of the world, but not necessarily all relations in the database. This can be achieved by creating suitable derivatives. For example, assume that a relation R of a given derivative, implemented as 4QL module D, is to be closed subject to some conditions. This can be expressed by the following rule of a 4QL module other than D, where $C(X)$ marks those conditions:

$$-R(X) \text{ :- } C(X), D.R(X) \text{ IN } \{\text{UNKNOWN}, \text{FALSE}\}. \tag{3}$$

For example, one may locally close $S(X,healthy)$ assuming:

$$-S(X,healthy) \text{ :- } Cl(X,moderate), S(X,healthy) \text{ IN } \{\text{UNKNOWN}, \text{FALSE}\}.$$

5.2.2 Default Reasoning

Another major technique of non-monotonic reasoning is default reasoning, intensively studied by numerous authors (see, e.g., [4, 5, 15, 18] and references there). Default rules have the form:

$$prerequisite : justification \vdash consequent, \tag{4}$$

with the intuitive meaning "deduce *consequent* whenever *prerequisite* is true and *justification* is consistent with current beliefs".

Assuming that *consequent* is an arbitrary literal, *justification* is a literal of a derivative implemented as module D and *prerequisite* is a conjunction of literals $P1, \ldots Pn$ of D, rules of the form (4) can be translated into:

$$\begin{aligned} consequent \text{ :- } & D.P1 = \text{TRUE}, \ldots, D.Pn = \text{TRUE}, \\ & D.justification \text{ IN } \{\text{TRUE}, \text{UNKNOWN}\}. \end{aligned}$$

For example, when noise level is moderate and pollution level is not known to be low, the situation is by default not healthy:

$$N(a, moderate) : \neg P(a, low) \vdash \neg S(a, healthy),$$

which can be expressed in 4QL as:

$$-S(a, healthy) \text{ :- } N(a, moderate) = \text{TRUE}, \ \neg P(a, low) \text{ IN } \{\text{TRUE}, \text{UNKNOWN}\}.$$

5.2.3 Autoepistemic Reasoning

Autoepistemic reasoning [20] uses rules of the form:

$$\text{"If you do not know } R\text{, conclude } \neg R\text{."} \tag{5}$$

If R is a literal, formulas of the form (5) can be translated into 4QL rules of the form:

$$-R \text{ :- } M.R \text{ IN } \{\text{UNKNOWN}\}.$$

We can also extend this technique by deriving conclusions on the basis of the lack of knowledge of other agents: "if an agent does not know R, conclude $\neg R$."

For example, one may express the following simple rule:

$$-S(X, healthy) \text{ :- } D.S(X, healthy) \text{ IN } \{\text{UNKNOWN}\}.$$

5.2.4 Defeasible Reasoning

Defeasible reasoning is another form of nonmonotonic reasoning, aiming at resolving inconsistencies [8]. A rule-based form of defeasible reasoning has been introduced in [21]. Rules have the form similar to 4QL, but the underlying semantics is two-valued. Possible inconsistencies are resolved by placing priorities on rules.

Such priorities can easily be modeled in 4QL [16]. For example, consider the following defeasible rules in a module D, where rule (7) has a higher priority than (6):

$$S(X,healthy) \quad :\!- P(X,low),N(X,low). \tag{6}$$
$$-S(X,healthy) :\!- Cl(X,pollutive). \tag{7}$$

The following rules resolve possible inconsistencies according to the above priority:

$$S(X,healthy) \quad :\!- \quad D.S(X,healthy) \text{ IN } \{\text{TRUE}\}.$$
$$-S(X,healthy) \quad :\!- \quad D.S(X,healthy) \text{ IN } \{\text{INCONS}, \text{FALSE}\}.$$

6 Conclusions

In the paper we have presented a novel framework for engineering beliefs which for the first time enjoys the following properties:

- heterogeneity of agents' epistemic profiles;
- natural handling of inconsistencies and gaps in beliefs;
- fusion of distributed beliefs performed in a highly distributed manner;
- flexibility of defining belief fusion rules;
- tractability, making practical applications feasible.

We have also indicated the implementation tool 4QL, which allows implementing epistemic profiles constructible in deterministic polynomial time. The epistemic profiles create conceptual bases for belief structures. They can be constructed and queried in a tractable manner.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley Pub. Co (1996)
2. Alferes, J.J., Moniz Pereira, L.: Reasoning with Logic Programming. LNCS, vol. 1111. Springer, Heidelberg (1996)
3. Belnap, N.: A useful four-valued logic. In: Eptein, G., Dunn, J. (eds.) Modern Uses of Many Valued Logic, pp. 8–37. Reidel (1977)
4. Besnard, P.: An Introduction to Default Logic. Springer (1989)
5. Brewka, G.: Non-Monotonic Reasoning: Logical Foundations of Commonsense. Cambridge University Press (1991)
6. Cadoli, M., Eiter, T., Gottlob, G.: Complexity of propositional nested circumscription and nested abnormality theories. ACM Trans. Comput. Log. 6(2), 232–272 (2005)
7. Cadoli, M., Schaerf, M.: A survey on complexity results for non-monotonic logics. Journal Logic Programming 17, 127–160 (1993)
8. Damásio, C., Pereira, L.: A survey of paraconsistent semantics for logic programs. In: Gabbay, D.M., Smets, P. (eds.) Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol. 2, pp. 241–320. Kluwer (1998)
9. Dubois, D.: On ignorance and contradiction considered as truth-values. Logic Journal of the IGPL 16(2), 195–216 (2008)

10. Dunin-Kęplicz, B., Szałas, A.: Epistemic Profiles and Belief Structures. In: Jezic, G., Kusek, M., Nguyen, N.-T., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2012. LNCS, vol. 7327, pp. 360–369. Springer, Heidelberg (2012)
11. Eiter, T., Gottlob, G.: Propositional circumscription and extended closed-world reasoning are Π_2^P-complete. Theoretical Computer Science 114(2), 231–245 (1993)
12. Etzioni, O., Golden, K., Weld, D.: Sound and efficient closed-world reasoning for planning. Artificial Intelligence 89, 113–148 (1997)
13. Fages, F.: Consistency of Clark's completion and existence of stable models. Methods of Logic in Computer Science 1, 51–60 (1994)
14. Gottlob, G.: Complexity results for nonmonotonic logics. Journal of Logic and Computation 2(3), 397–425 (1992)
15. Łukaszewicz, W.: Non-Monotonic Reasoning - Formalization of Commonsense Reasoning. Ellis Horwood (1990)
16. Małuszyński, J., Szałas, A.: Living with Inconsistency and Taming Nonmonotonicity. In: de Moor, O., Gottlob, G., Furche, T., Sellers, A. (eds.) Datalog 2010. LNCS, vol. 6702, pp. 384–398. Springer, Heidelberg (2011)
17. Małuszyński, J., Szałas, A.: Logical foundations and complexity of 4QL, a query language with unrestricted negation. Journal of Applied Non-Classical Logics 21(2), 211–232 (2011)
18. Marek, V., Truszczyński, M.: Nonmonotonic Logic. Springer (1993)
19. Mascardi, V., Demergasso, D., Ancona, D.: Languages for programming BDI-style agents: an overview. In: Corradini, F., De Paoli, F., Merelli, E., Omicini, A. (eds.) WOA 2005 - Workshop From Objects to Agents, pp. 9–15 (2005)
20. Moore, R.: Possible-world semantics for autoepistemic logic. In: Proc. 1st Nonmonotonic Reasoning Workshop, pp. 344–354 (1984)
21. Nute, D.: Defeasible logic. In: Handbook of Logic in Artificial Intelligence and Logic Programming, pp. 353–395 (1994)
22. Szałas, A.: How an agent might think. Logic Journal of the IGPL (to appear, 2012)
23. Vitória, A., Małuszyński, J., Szałas, A.: Modeling and reasoning with paraconsistent rough sets. Fundamenta Informaticae 97(4), 405–438 (2009)

A Multi-tiered Recommender System Architecture for Supporting E-Commerce

Luigi Palopoli, Domenico Rosaci, and Giuseppe M.L. Sarné

Abstract. Nowadays, many e-Commerce tools support customers with automatic recommendations. Many of them are centralized and lack in efficiency and scalability, while other ones are distributed and require a computational overhead excessive for many devices. Moreover, all the past proposals are not "open" and do not allow new personalized terms to be introduced into the domain ontology. In this paper, we present a distributed recommender, based on a multi-tiered agent system, trying to face the issues outlined above. The proposed system is able to generate very effective suggestions without a too onerous computational task. We show that our system introduces significant advantages in terms of openess, privacy and security.

1 Introduction

Nowadays, a large number of *Recommender System*s (RSs) is used to promote e-Commerce (EC) activities [11] but often they fall when transactions occur between customers and merchants (B2C), mainly for a inadequate exploration of the market space, ineffective communications between the actors and for lack of security and privacy in the transactions. To solve such issues, new B2C systems, characterized by high levels of automation, exploit *software agents* that, acting on the customers' behalf, allow a B2C transaction to be carried out without human intervention.

A RS provides his user with potentially useful suggestions for his purchases [14] based on a representation of his interests and preferences across the phases of a B2C transaction. Different behavioural models describe such phases, as the well known *Consumer Buying Behaviour* (CBB) model [6] base on six stages, namely: *i) Need*

Luigi Palopoli
Università della Calabria, 87036 Rende (CS) (Italy)
e-mail: palopoli@deis.unical.it

Domenico Rosaci · Giuseppe M.L. Sarné
Università Mediterranea, 89122 Reggio Calabria (Italy)
e-mail: {domenico.rosaci,sarne}@unirc.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 71–81.
springerlink.com

Identification; *ii) Product Brokering*; *iii) Merchant Brokering*; *iv) Negotiation*; *v) Purchase and Delivery*; *vi) Service and Evaluation*. Software agents [16] are usually exploited to monitor a user during these stages for building his profile. The RSs present in the EC sites can adopt centralized or distributed architecture. The first generate suggestions only on the server side but their performances are lacking in terms of efficiency and scalability and customers' privacy (due to the centralization of personal information), and this potentially affects the quality of their suggestions. The alternative approach implies distribution [9, 10] but its complexity could generate unacceptable computational overheads on the client as, for instance, a mobile device. Moreover, existing RSs assume homogeneous system components, implying that it is difficult for the users to add personal knowledge in the system.

In this paper, we present a **D**istributed **A**gent **R**ecommender for **E**-**C**ommerce (**DAREC**) based on a multi-tiered agent system. It allows to *i)* increase the distribution degree of the RS, *ii)* generate effective recommendations without any onerous computational task on the client side, *iii)* introduce significant advantages in openess and privacy. The basic idea of this framework (see Figure 1) is that each customer is assisted by three software agents, each of which, autonomously of the other agents, deals with a different CBB stage . Each agent runs on a different thread in the customer's client and this improves the efficiency of the overall process being agent interactions *specialized*. Each customer's agent, during its activity, can interact with the DAREC sellers' sites distributed over the Internet, where each seller site is assisted by a seller agent provided with both a *product catalogue* and *customers' profiles* encoding the preferences of each customer that visited the site in the past. This interaction allows the customer agent to generate content-based (CB) recommendations for the customer and also makes the site able to generate personalized presentations of the products for its visitors to support the site visit. The agents also interact with the seller agents and reciprocally generate collaborative filtering (CF) recommendations. This way if a customer c_1 needs to interact with a customer c_2 for need identification purposes, his NI-agent simply interacts with the c_2's NI-agent. The other agents of c_1 and c_2 are free to perform other activities improving the system performances with respect to those systems where a unique customer's agent can execute only one activity at time.

The remaining of the paper is organized as follows. In Section 2 we introduce the knowledge representation exploited in DAREC, while Section 3 describes the agents' behaviour. Section 4 deals with some related work. Section 5 presents some experiments to evaluate our proposal and in Section 6 we draw our final conclusions.

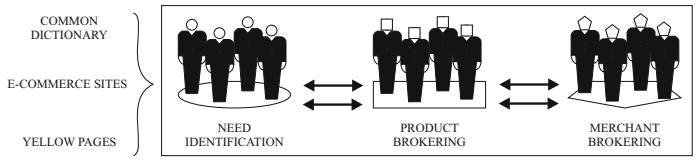


Fig. 1 The DAREC Architecture

2 The DAREC Knowledge Representation

The DAREC community shares a common dictionary storing the names of basic product categories of interest and their reciprocal relationships. Moreover, each customer's agent profile encodes all the information necessary to manage its CBB stage. Similarly, each seller's agent manages a catalogue of offered products organized in categories. Finally, in order to allow a collaboration between agents the information stored in a "yellow page" data structure are exploited (see below).

More in detail, a *Category Dictionary* $\mathscr{D}$ contains (*i*) a set $\mathscr{D}_{\mathscr{C}}$ of *product categories* and (*ii*) a set of *category links* $\mathscr{D}_{\mathscr{R}}$ between categories, denoted by $\langle cat_1, cat_2, t\rangle$ where cat_1, $cat_2 \in \mathscr{D}$ and t is the *type* of the link that can be: *i) isa-linked*, denoted $\langle cat_1, cat_2, ISA\rangle$, iff all the products belonging to cat_1 also belong to cat_2 ; *ii) synonymy-linked*, denoted $\langle cat_1, cat_2, SYN\rangle$, iff both all the products belonging to cat_1 also belong to cat_2 and vice versa; *iii) overlap-linked*, denoted $\langle cat_1, cat_2, OVE\rangle$, iff there exist some product of cat_1 that also belong to cat_2, and viceversa. Note that if two categories are synonymy-linked, they are also overlap-linked; *iv) commercial-linked*, denoted $\langle cat_1, cat_2, COM\rangle$ iff we suppose that the customers usually purchase both products belonging to cat_1 and cat_2.

We represent a category dictionary $\mathscr{D}$ (called COMMON and publicly available) as a direct labeled graph $G(\mathscr{D}) = \langle \mathscr{D}_C, \mathscr{D}_R\rangle$, where; for each category $cat \in \mathscr{D}_C$ there is a node called $name_{cat}$ associated with a label denoted by $info_{cat}$; for each arc $\in \mathscr{D}_R$ there is a link $\langle cat_i, cat_j, t\rangle$ oriented from cat_i to cat_j and labeled by t. Nodes represent product categories, which the products offered by the sellers belong to, and arcs represent existing relationships between categories. Moreover, we say that cat_i and cat_j belong to a *category relationship* (ISA($\mathscr{D}$), SIN($\mathscr{D}$), OVE($\mathscr{D}$) or COM($\mathscr{D}$)) if they are in the same dictionary and the relationships $\langle cat_i, cat_k, t\rangle$ and $\langle cat_k, cat_j, t\rangle$ belong to a more general (ISA, SIN, OVE or COM)-relationship, while they are called *generally related*, denoted by GEN($\mathscr{D}$), if there is in $\mathscr{D}$ a path between their associated nodes independently of the specific arc-labels.

2.0.1 The Personal and the Site Profiles

In a DAREC community ($\mathscr{C}$), in order to handle the Need Identification (resp. Product Brokering, Merchant Brokering) CBB stage, each customer $c \in \mathscr{C}$ is assisted in that stage by an agent, called NI_c (resp. PB_c, MB_c). Each agent stores in a profile, called NI (resp. PB, MB)-profile all the c's information necessary to handle the associated CBB stage. The profile is implemented by a category dictionary $\mathscr{P}(NI_c)$ (resp. $\mathscr{P}(PB_c)$, $\mathscr{P}(MB_c)$) such that its nodes represent categories of interest (resp. product categories relative to suitable products or merchants) for c, arcs represent links between categories and each category (resp. product of interest, merchant) is associated with a quantitative evaluation of the c's interest. Moreover, since a product (resp. merchant) search could be not activated for each category (resp. product) of interest for c, the categories belonging to $\mathscr{P}(PB_c)$ (resp. $\mathscr{P}(MB_c)$) are generally a subset of those in $\mathscr{P}(NI_c)$ (resp. $\mathscr{P}(PB_c)$). As a consequence, each arc of the PB (resp. MB)-profile, between cat_i and cat_j is a copy of the corresponding arc

belonging to the NI (resp. PB)-profile (included the label). Finally, each category belongs to either the common dictionary or to a "personal" customer's category (understandable to the other agents being in a general relationship with at least another category of the common dictionary).

In DAREC, each seller $s \in \mathscr{S}$ is associated with a seller agent that stores in its site profile (SP_s) a catalogue of the offered products and some information about the preferences of its past customers. Also SP_s is represented by a category dictionary whose nodes represent product categories in which s offers products and whose arcs represent relationships between categories. We introduce the mapping $SP_s(cat)$, that accepts the category cat as input and returns the tuple $SP_s(cat) = \langle prod, customers \rangle$, where $prod$ is a list of products that s offers in the category cat. To access the elements of this list, we use a mapping $SP_s(cat).prod(p)$ that accepts as input a product p and returns the tuple $SP_s(cat).prod(p) = \langle price, payment, format \rangle$, such that $price$ is price of p, $payment$ is the set of payment methods available for p and $format$ is the format available for p. The $price$ mean depends if the price is fixed or it is the reserved price in an "auction". Finally, the element $customers$ is a list of customers interested in products of the category cat that, for each customer, stores a list of those products of the site which the customer is interested in.

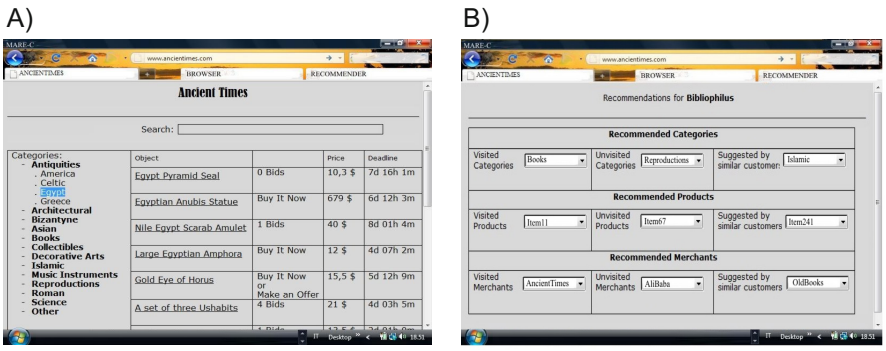


Fig. 2 A)An example of a DAREC site; B)The recommendations provided by the personal agents

2.0.2 The Yellow Pages YP

This data structure allows customers of the DAREC community to publish their interests, in order to facilitate mutual collaboration. YP is a set of category dictionaries $\{YP_c\}$, one for each customer c. In particular, YP_c is a sub-graph of c's NI-profile, containing those categories $cat \in \mathscr{P}(NI_c)$ such that $NI_c(cat).i = public$.

3 Agent's Behaviour in DAREC

The agent running on the client used by the customer c to visit the DAREC EC sites (i) supports the Web site navigation like a normal browser and (ii) generates

suggestions for the user. To this purpose, the client interface is provided with the functionalities *Browser* and *Recommender* described below. If c is a newcomer he should build an initial NI-Profile $\mathscr{P}(NI_c)$ by adding the categories of his interest, together their relationships, from the dictionary COMMON. Moreover, c could add to $\mathscr{P}(NI_c)$ some personal category $cat \notin COMMON$, by specifying its name and path in $\mathscr{P}(NI_c)$ joining cat with at least a category $cat^* \in COMMON$. Besides, for each selected category c should specify his interest degree $NI_c(cat).i$ and the visibility mode $NI_c(cat).mode$.

3.1 The Browser: Agents Working "over the Shoulders"

Each DAREC site allows the customer c *i)* to "navigate" through the categories by clicking on the tab "Browser" (the *Categories* list is on the left in Figure 2-A) or *ii)* to use the *Search* tool to perform a keyword-based search among the products sold at fixed price (i.e., *Buy-It-Now*) or with an auction (i.e., *Make an Offer*). For each product p, belonging to a category cat, c can perform the actions of: *(A1)* selecting the product for examining the offer; *(A2)* watching the product; *(A3)* purchasing the product. Each action A1, A2 or A3 performed by c implies a call to the agents NI, PB and MB that automatically update their profiles and in particular:

- The NI-agent is called, feeding it the category cat. If cat is absent in its profile, it is added therein and its interest value $NI_c(cat)$ is set to an initial value $iniInt$. Then the NI-agent requires to the PB and MB agents to add cat and its interest value to their profiles. Otherwise, if $cat \in$ NI-profile, its interest value is updated to $NI_c(cat) = min(1, NI_c(cat) + \Delta_a)$, where $\Delta_a \in [0,1]$ (with $a = A1, A2, A3$) it is arbitrarily set by c to weight the performed action. The value $NI_c(cat)$ is then passed to the agents PB and MB for updating their profiles.
- The client calls the PB-agent to pass the product p. If $p \notin$ PB-profile then it is added to the list $PB_c(cat).prod$ with $iniInt$ as interest value and their insertion in the the list $MB_c(cat).prod \in$ MB-agent is required. Otherwise, if $p \in$ PB-profile its interest value is updated to $PB_c(cat).prod(p).i = min(1, PB_c(cat).prod(p).i + \Delta_a)$ and passed to MB to be copied in $MB_c(cat).p.i$.
- The client calls the MB agent, passing the seller s. If $s \notin$ MB-profile, then it is added to the list $MB_c(cat).sellers$ with $numT = 1$ and a score of $iniInt$. Otherwise, $numT$ increased by 1 and the score is updated to $MB_c(cat).sellers(s).ev = min(1, MB_c(cat).sellers(s).ev + \Delta_a)$.

Periodically, the $NI_c(cat).i$ (resp. $PB_c(cat).prod(p).i$, $MB_c(cat).sellers(s).ev$) value associated with the NI (resp. PB, MB)-profile, after a τ_{NI} (resp. τ_{PB}, τ_{MB}) time period passed from its last update, is decreased of ρ_{NI} (resp. ρ_{PB}, ρ_{MB}), a c's parameter ranging in $[0,1]$. Also the seller agent of s updates its list $SP_s(cat).customers \in SP_s$ after each customer's action that involves a product $p \in cat$. In particular, if $c \notin SP_s(cat).customers$ a new element $SP_s(cat).customers(c)$ is added to it. Moreover, p is added to the list $SP_s(cat).customers(c).prod$ and the number of transactions $SP_s(cat).customers(c).numT$ is increased. Otherwise, if $c \in SP_s(cat).customers$, the

seller agent updates the element $SP_s(cat).customers(c)$ by increasing the number of transactions $SP_s(cat).customers(c).numT$ and inserting p in $SP_s(cat).customers(c).prod$ if it is absent.

3.2 The Recommender: Exploiting Customer's Profiles

When c selects the tab "Recommender" in his client, then some suggestions are generated for him and visualized in a page having a section for each supported stage (Figure 2-B). Suggestions are generated by the each agent in a "cascade" mode, i.e. firstly the customer chooses a category from those in section "Recommended Categories", then a set of products is suggested in section "Recommended Products" and chosen a product, a set of merchants selling that product is suggested in section "Recommended Merchants".

The **NI-agent** suggests to c a set of categories visualized in the client section "Recommended Categories" in the three distinct list-boxes (Figure 2-B):

- **Visited Categories** contains categories selected with a CB approach from the NI-profile $\mathscr{P}(NI_c)$ built by monitoring the c's activity (see Section 3).
- **Unvisited Categories** lists categories unknown to c, but considered interesting for him by his NI-agent. This agent uses a *relationship-based* mechanism to exploit the interaction between the c's NI-agent and the agent of each site that he visited in the past. In particular, the agent of a seller s, for each category cat visited by c (i.e. $c \in SP_s(cat).customers$), determines all the categories $cat^* \in SP_s$ such that cat^* is unvisited by c and there exists a path in SP_s between cat and cat^*; these categories are sent to the c's NI-agent, to be added to this list.
- **Suggested by Similar Customers**, where the categories are determined, with a CF technique, on the whole EC customer's navigation history by the c's NI-agent collaborating with the NI-agents of customers similar to c for interests. To this aim, it is exploited the public repository YP (see Section 2) storing, for each DAREC customer x his public interest profile YP_x. The c's NI-agent computes the similarity degree $s(c,x)$ between its profile $\mathscr{P}(NI_c)$ and each YP_x by using the Jaccard measure of the set of nodes of $\mathscr{P}(NI_c)$ and YP_x for n customers (a parameter set by c), that is $s(c,x) = |NODES(\mathscr{P}(NI_c)) \bigcap NODES(YP_x)| / |NODES(\mathscr{P}(NI_c)) \bigcup NODES(YP_x)|$, where NODES(G) returns the set of nodes of its input graph. Then, the c's NI-agent determines, for each similar customer x, those categories stored in the public profile $YP_x \notin \mathscr{P}(NI_c)$, and adds them to this list.

The **PB-agent** (resp. **MB-agent**) suggests, in the section "product recommendations" (resp. "merchant recommendations") a set of products (resp. sellers) belonging to a category cat selected by c from the recommended categories (resp. 'products) on his client. These products are visualized in the listboxes:

- **Visited Products (resp. Merchants)**, it contains products (resp. merchants) of the category $cat \in$ PB-profile $\mathscr{P}(PB_c)$ (resp. MB-profile $\mathscr{P}(MB_c)$), ordered by score.
- **Unvisited Products (resp. Merchants)**, this list is built by exploiting a collaboration between the c's PB (resp. MB)-agent and the seller agent of each site s the

customer c visited in the past. In particular, for a PB-agent each listed $p \in cat$ is unvisited by c and belongs to each site profile, while for the MB-agent the set is formed by those sellers having *cat* in their profiles and are unvisited by c in the past.
• **Suggested by Similar Customers**, these suggestions are based on the collaboration between the PB (resp. MB)-agents of c and of other customers similar to him for interests (their list is provided by the c's NI-agent). Each of the PB (resp. MB)-agent of these customers sends to c's PB (resp. MB)-agent its set of products $PB_x(cat).prod$ (resp. merchants $MB_x(cat).sellers$) that is added to this list. The PB (resp. MB)-agent shows the products (resp. merchants) belonging to the listbox "Visited Products" (resp. "Visited Merchants"), ordered by value, and the products (resp. merchants) belonging to the listboxes "Unvisited Products" (resp. "Unvisited Merchants") and "Suggested by Similar Customers", ordered alphabetically.

Each **seller agent** SA_s associated with a seller s exploits its profile SP_s to personalize the site presentation for the each customer c that is visiting it. When c returns to visit the site, the element $SP_s(cat).customers(c)$ already exists in the SA_s profile. Using such information, SA_s personalizes its home page for c by visualizing in a "Shop Window" all the $p \in SP_s(cat).customers(c).prod$, ordered by interest value. SA_s uses this list as a sort of *local profile* related to c and at the same time, it increments the value $SP_s(cat).customers(c).numV$ to consider his current visit. Otherwise, for the first time c's visit the default home page is visualized.

4 Related Work

Centralized RSs are widely used in EC Web sites, for example, the *Amazon* site [1] adopts some recommender tools based on the customer's browsing history, past purchases and purchases of other customers. The system drives a customer to buy something because this is related to something that he purchased before, or because this is popular with other customers. Another case is that of the RSs embedded in auction sites [4], as in eBay [5], that generates recommendations using its feedback profile features. Among the centralized approaches in [9] the authors propose a system which stresses on freshness, novelty, popularity and limitedness in product recommendations. All these centralized approaches generate, similarly to DAREC, both CB and CF recommendations and store all the private information about customers and sellers necessary to generate recommendations. Differently to DAREC is that they are not open and use pre-defined dictionaries of terms for defining the product categories while DAREC allows the users to define new terms in their personal ontologies. Distributed recommender systems (DRSs), increased in diffusion in these later years, share information and computation tasks among more entities. DRSs are more critical in design and performances optimization [8] than centralized RSs but *(1)* promise scalability in time and space complexities, *(2)* avoid the failure risks due to a central database running on a unique server and *(3)* preserve privacy and security. DSRs often adopt peer-to-peer (P2P) and agents technologies to easily *i)* exchange data locally stored on each peer (e.g., Chord [13]) reducing the task to locate a specific resource and *ii)* provide communication, cooperation and

negotiation rules, respectively. In a mall, adaptive learning agents associated with each shop in CASy [3] process shop transactions and analyze information about interests of customers entering the shop that are derived by his profile, keywords, product queries and by other shop agents for proposing to the consumer suitable suggestions in an efficient and adaptive way. A multi-agent system (MAS) implementing a knowledge-based DSR applied to the tourism domain is discussed in [7]. Agents cooperate to suggest travel packages to a user. They are expert in specific domains (hotel, flights, interchanges, etc) and autonomously select the most suitable sub-task of a recommendation request to deal with their competence. Their suggestions are then composed in a final recommendation. All the referred DRSs take advantage of the distributed architectures in terms of scalability, risks failure, privacy and security. Differently from DAREC, none of them deal with the Need Identification, Product or Merchant Brockering phases based on a description of the consumer's interests and preferences (with the partial exception of [2] that can also avoid the use of such a profile). All the described recommenders and DAREC adopt (or can easily adopt, as [12, 15]) agent-based and/or P2P systems to find similar neighbors, resources and exploit predefined services. Agent specialization is introduced in [7] but it is relative just to the item typology, while DAREC is based on agent communities, where each agent is specialized in a different EC phase (as described in the CBB model). A similar concept is also present in [15] but it involves a set of recommenders, each one linked to a different organizations and having a particular point of view in generating suggestions.

5 Efficiency and Effectiveness of DAREC

In this Section, we discuss efficiency and some experimental results about the advantages of our approach. In terms of efficiency, for a community of n customers and m sellers, a unique centralized agent managing all the three phases has computational cost of $NC = \Sigma_{i=1}^{3} n_i \cdot k_i$, where k_i and n_i are the number of contemporary sessions activated and of operations needed for a user to manage the Need Identification, Product and Merchant Brokering phases, respectively. Instead, in DAREC, for a given CBB stage, each user's agent can deal with more different issues running on different threads. Let α_i, be the multi-threading degrees for a specific CBB stage and let $\beta = k_1 + k_2$ be the computational overhead due to the communications between the local agents (i.e., the Need Identification agent calls the Product Broker agent, that in its turn can call the Merchant Broker agent that does not run any additional operation). In this way the computational cost for a CPU will be $ND = \beta + \Sigma_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i$ and the computational advantage (ρ), due to the distribution in DAREC, is equal to $\rho = (\beta + \Sigma_{i=1}^{3} \alpha_i \cdot n_i \cdot k_i) / \Sigma_{i=1}^{3} n_i \cdot k_i$ where if, for simplicity, $\alpha = \alpha_1 = \alpha_2 = \alpha_3$, $k = k_1 = k_2 = k_3$ and $N = n_1 + n_2 + n_3$, the above formula becomes $\rho = \alpha + 2/N$. Therefore, the advantage of using DAREC is perceivable with a small multi-threading contribution (i.e. high values of α) in presence of a reasonably high number of operations (i.e. an intense EC activity), while for a high multi-threading activity the advantage shows up also for small N values.

In terms of effectiveness, the time exploited to perform B2C processes in serial and multi-threading way has been compared by means of a software appositively designed. To this aim, we considered a period of 2 hours where a set of 500 customers finalize all their B2C processes with a purchase dealing with a merchant population (M) of 10 units. Moreover, the merchant has to satisfy also the requests due to other customers that could absorb significant merchants' servers resources. This is taken into account by means of an overhead (O) of $1 \div 100$ requests for second, randomly shared among the merchants. Finally, a lot of different of communication, computational and behavioural parameters have been tuned to model realistic B2C processes. Obviously, in order to compute in average the time (in seconds) needed to perform a purchase process in a multi-threading (T_m) and in a serial (T_s) modality the same values for the parameters have been used. More in detail, T_m (i.e. T_s) has been computed as $T_m = \sum_{i=1}^{NP} T_m/NP$, where NP is the number of purchases, randomly fixed, performed in the considered test session.

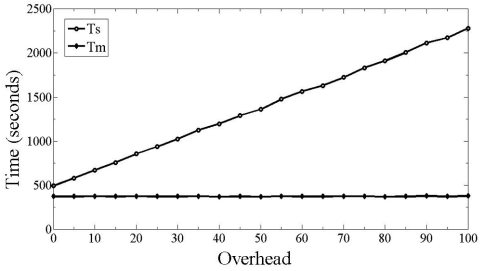


Fig. 3 The average serial (T_s) and multi-threading (T_m) times (in sec.) needed to carry out a B2C process depending on the Overhead by considering 500 Customers and 10 Merchants

The experimental results shown in Fig. 3 confirm that the DAREC approach consumes in average about the 25% of time less then the serial approach in performing a purchase in absence of overhead and when the overhead grows also T_s grows with it while T_m is almost uniform. In Table 1 are reported the average gain (G) in percentage of T_m with respect to T_s for different values of the overhead. This behavior is due to the fact that changes in the number of merchants, overheads and so on, have a minimal impact on T_m and very high impact on T_s. This because, in average, each merchant's server is busy to satisfy the customers' requests and T_s grows with the level of "saturation" of the merchants' servers worsening the quality of their service.

Table 1 The average gain ($G = T_s/T_m$) to carry out a B2C process depending on the number of Overhead by considering 500 Customers and 10 Merchants

O	0	5	10	20	30	40	50	60	70	80	90	100
G	24.61	36.12	44.63	56.43	63.98	69,19	73,05	76,39	78,39	80,75	82,28	83,61

6 Conclusions

In this paper, we have presented advantages and limitations of the DAREC distributed architecture that introduces novel, original characteristics with respect to other recommenders. DAREC allows to the different CBB stages of an EC process to be assigned to a different agent creating a tier of specialized agents. This architecture reduces the computational cost for the device on which the local agents run and the presence of specialized agents improves the users' knowledge representations, the openness of the system and the privacy degree.

Acknowledgements. This work was partially funded by the Italian Ministry of Research through the PRIN Project "Entity Aware Search Engines" and by the DEIS (Università della Calabria).

References

1. Amazon (2011), http://www.amazon.com
2. Awerbuch, B., Patt-Shamir, B., Peleg, D., Tuttle, M.R.: Improved Recommendation Systems. In: Proc. of 16th ACM-SIAM Symp. on Discrete Algorithms, pp. 1174–1183. SIAM (2005)
3. Bohte, S.M., Gerding, E., La Poutré, J.A.: Market-based Recommendation: Agents that Compete for Consumer Attention. ACM Trans. Internet Techn. 4(4), 420–448 (2004)
4. Culver, B.: Recommender System for Auction Sites. J. Comput. Small Coll. 19(4), 355–355 (2004)
5. eBay (2011), http://www.ebay.com
6. Guttman, R.H., Moukas, A., Maes, P.: Agents as Mediators in Electronic Commerce. Electronic Markets 8(1), 22–27 (1998)
7. Lorenzi, F., Correa, F.A.C., Bazzan, A.L.C., Abel, M., Ricci, F.: A Multiagent Recommender System with Task-Based Agent Specialization. In: Ketter, W., La Poutré, H., Sadeh, N., Shehory, O., Walsh, W. (eds.) AMEC 2008. LNBIP, vol. 44, pp. 103–116. Springer, Heidelberg (2010)
8. Olson, T.: Bootstrapping and Decentralizing Recommender Systems. Ph.D. Thesis, Dept. of Information Technology. Uppsala Univ. (2003)
9. Parikh, N., Sundaresan, N.: Buzz-based Recommender System. In: Proc. of 18th Int. Conf. on World Wide Web (WWW 2009), pp. 1231–1232. ACM (2009)
10. Rosaci, D., Sarné, G.M.L., Garruzzo, S.: MUADDIB: A Distributed Recommender System Supporting Device Adaptivity. ACM Trans. Inf. Syst. 27(4) (2009)
11. Schafer, J.B., Konstan, J.A., Riedl, J.: E-Commerce Recommendation Applications. Data Min. Knowl. Discov. 5(1-2), 115–153 (2001)
12. Schifanella, R., Panisson, A., Gena, C., Ruffo, G.: MobHinter: Epidemic Collaborative Filtering and Self-Organization in Mobile Ad-Hoc Networks. In: Proc. of ACM Conf. on Recommender Systems (RecSys 2008), pp. 27–34. ACM (2008)
13. Stoica, I., Morris, R., Karger, D.R., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. of SIGCOMM 2001, pp. 149–160 (2001)

14. Wei, K., Huang, J., Fu, S.: A Survey of E-Commerce Recommender Systems. In: Proc. of 13th Int. Conf. on Service Systems and Service Management, pp. 1–5. IEEE (2007)
15. Weng, L.-T., Xu, Y., Li, Y., Nayak, R.: A Fair Peer Selection Algorithm for an e-Commerce-Oriented Distributed Recommender System. In: Proc. of 2006 Conf. on Adv. in Intell. IT, pp. 31–37. IOS (2006)
16. Wooldridge, M., Jennings, N.R.: Agent Theories, Architectures, and Languages: A Survey. In: Wooldridge, M.J., Jennings, N.R. (eds.) ECAI 1994 and ATAL 1994. LNCS, vol. 890, pp. 1–39. Springer, Heidelberg (1995)

Healthcare Interoperability through a JADE Based Multi-Agent Platform

Miguel Miranda, José Machado, António Abelha, and José Neves

Abstract. In the healthcare arena and in the design of its information systems, a fundamental principle is required for providing a better service throughout all departments, that principle is interoperability. Establishing connections among distinct service providers can be complicated and very often result in complex mesh of end-to-end flow of information which is too often hard to maintain and strongly coupled. Multi-agent system technology is a strong technology to address this subject. In this paper it is described an architecture of a multi-agent system, which aims to provide to the Health Information System with a distributed and consolidated tool towards implementing loosely-coupled interoperability among its systems.

1 Introduction

Due to the specificities of each of its areas and services provided, the healthcare environment is architecturally composed by an intricate set of information systems. Under these circumstances distinct solutions must share data and information consistently and as a whole. The communication among them is of the essence for the optimisation of existing resources and the improvement of the decision making process through consolidation, verification and dissemination of information. Henceforth, within the healthcare environment the integration of all otherwise secluded applications is fundamental for the development of a scalable and functional Health Information System (HIS) [4]. The concept of a HIS defines it as the abstract entity that encompasses the group of integrated and interoperable solutions within the healthcare institution. It is henceforth essential to imbue any HIS with the capability to allow communication among different and otherwise secluded systems, avoiding their centralisation and dissemination of end-to-end connections, which restrict the dynamics within the IT infrastructure. The non-modularity of services adds complexity to alterations and improvements, increasing the global costs of the

Miguel Miranda · José Machado · António Abelha · José Neves
Universidade do Minho, CCTC, Braga, Portugal
e-mail: {miranda,jmac,abelha,jneves}@di.uminho.pt

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 83–88.
springerlink.com

information systems [1]. Therefore, it is understandable the present concern demonstrated by distinct international institutions, responsible for financing and regulating the purchase and development projects for new HIS, with matters of flexibility, interoperation and integration of heterogeneous systems [2].

In this paper it is described an architecture of a multi-agent system, which aims to provide to the HIS with a distributed and consolidated tool towards implementing loosely-coupled interoperability among its systems. Firstly, this paper approaches a general perspective on multi-agent systems towards interoperability, followed with a mildly detailed description of the architectures modules, its dependencies and methodologies. Its of relevance to mention that this architecture is in production state at the Oporto Hospital centre and provides an interoperability interface for most of the existing information services as well as consolidated knowledge-based from which a clinical health record solution is based on.

2 Multi-Agent Systems towards Interoperability

The multi-agent system (MAS) paradigm has been an interesting technology in the area of interoperability in healthcare. Being multi-agent architectures a field of research of distributed artificial intelligence, this technology is intrinsically connected to the basilar concepts that define a distributed architecture, while being distinct in the intrinsic definition of an agent versus the properties of the general middle-wares of many others distributed architectures.

Congruently with these concerns, present tendencies regarding research and industry in interoperability applied to healthcare information systems, indicate the potential of agent oriented architecture [3] [5]. Although healthcare standards like HL7 are completely distinct from agent communication standards, HL7 services can be also implemented under the agent paradigm.

3 Architectural Analysis

Considering the complexity and heterogeneity inherent to the creation of an interoperability architecture and modelling of information flow, in this section it is proposed an archetype for healthcare interoperability based on multi-agent systems. Furthermore, the interoperability architecture based on this archetype is detailed and explained as an consistent methodology towards addressing the problem at hand.

3.1 Archetype Data

The methodology proposed divides the application development in the modular cores, which are related with the perspective os software development: Core Agent Framework; Database Independent Persistence Module; Web-services Tier; Ontology and Terminology Server; and Agent Development Tier.

3.1.1 Core Agent Framework

This module uses JADE as the base agent development library and extends it adding capabilities to the agent which, were previously unavailable. Henceforth, this is not merely a module of reference for the JADE Framework, but rather an extension of several capabilities. Among these new features emphasis should be placed on continuous monitoring of activity and individual agent statistics; embedded hibernate features; persistent arguments and variables; environment configuration automation; and embedded internal persistent log monitor. Most of these features were achieved thought the extension of the jade.core.Agent class by a *aida.core.Agent* class which implemented several methods and used different modules.

Continuous Agent Monitoring

The continuous monitoring of activity enables statistics regarding the process time occupied by each agent in every container and platform in the CPU. This monitoring is of the essence for internal behaviours of the agent to know when its performance is limited by lack of CPU or internal memory. These statistics are stored in the persistent database so that a dynamic threshold can be calculated to start the agent migration to another container in a different machine.

Embedded Hibernate Features

The use of Hibernate implicates a creation of a Session Factory, which is responsible for the creation of a database connection in a safe and easy manner. From the perspective of the Agent Paradigm each agent is an individual, with independent life-cycle and behaviour, for this reason each agent must be capable to manage its own connections according to its own functioning.

Persistent Dynamic Arguments

This feature provides an active set of dynamic arguments which can persist through all states of and agent life cycle. These arguments are loaded at the creation of the agent and keep synchronous throughout its inner instructions. The persistence of these variables is extremely important for handling agent recovery to failure and to easy and dynamic agent arguments at boot level. These arguments are kept in a XML file stored in database, which is loaded and persisted through the inner behaviours of the agent.

Platform Configuration Automation

This extension of the Agent class enabled the definition of specific environment characteristics which are stored in the same tier that holds the distinct databases configuration.

Inner Persistent Log Monitor

A persistent monitor which enables ease of logging and also some automated registry of events in the persistent database was implemented in order to provide a more centralised information centre for the agent platform.

3.1.2 Database Interoperation Interface

There are several Object-Relational Mapping (ORM) frameworks available for Java. Several were analysed and evaluated in order to select one to be naturally integrated into the platform. This platform contains an hibernate oriented extension embedded in the core Agent class itself as persistence is considered a requirement for all agents.

Hibernate is considered an ORM library for Java, providing a framework for mapping an object-oriented domain model to a traditional relational database. This ORM aims to solve limitations and mismatch for data persistence shared among the object-oriented model and the relational databases model.

3.1.3 Web-Services Tier

Although agents can provide web-services quite easily, this option was considered as far too complex regarding the mobility and lifecycle the agent system must provide. Henceforth a web-service layer was independently created, so that the availability of this services could be ensured and improved.

3.1.4 Ontology and Terminology Server

This tier grants a agglomerated access to all needed ontologies and terminologies by the platform agents and external service providers. Within this centralised storage there are contained several healthcare standards and proprietary terminologies. Among the standard the most significant for interoperability is HL7 version 2 as it is in the field the most commonly used nowadays. The main aim of an interoperability archetype is to model and enable communication. Within this platform communication occurs among agents and non agent-based services. Default communication among agents is provided by its JADE core and follows all FIPA communications standards. In addiction in JADE there are numerous methodologies to implement agent ontologies through the usage of existing terminologies.

3.1.5 Agent Development Tier

On top of all previous layers stands the agent development tier (ADT), which depends directly or indirectly from all others and gathers the specific agent development. As it stands on top of all other tiers it is obvious it allows that all agents are provided with the same resources and basic characteristics. In different words this tier encompasses the individual agent types and embedded behaviours of each agent, while maintaining the internal general behaviours.

3.2 Architectural Concepts and Methodologies

This mentioned architecture is oriented towards the concepts of dissemination and consolidation of HIS information in order to synergistically improve the quality of all information system involved.

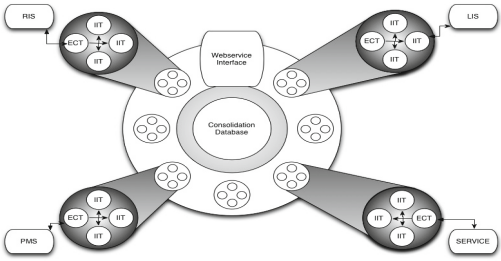


Fig. 1 Overall archetype of the interoperability platform

Within an HIS there are systems that can be considered as essential and which characteristics and information exchanged is rather standard. That is the case of the Patient Management System (PMS), the Radiological Information System (RIS) and the Laboratory Information System (LIS). However, all systems interact with each other indirectly and trough the communication with the agents within the MAS. Each system interacts with its specific designed agent type as in Figure **??**, however all systems communicate with each other through the internal communication of the multi-agent system. By avoiding end-to-end connections among systems we attain a higher level of looseness coupling.

4 Conclusions

The usage of multi-agent systems in interoperability problems constitutes a significant research opportunity to improve the communication among heterogeneous systems. Several of the research interests of agent technology such as ontologies, mobility and fault tolerance among many others can be of great use and interest to be applied in this area. The most important characteristic of this architecture and model is that instead of a mesh of end-to-end system communication or a major centralisation of processing, this paradigm is by nature distributed but allows a consolidation of processual and clinical validation of information. This consolidation is of the essence for the the establishment of a complete electronic health record in an environment in which heterogeneous information systems exist. The platform is now being used to resolve real time interoperability problems in several portuguese hospitals.

Acknowledgements. This research was performed with the support of the Portuguese Foundation for Science and Technology, with the grant SFRH/BD/65023/2009.

References

1. Aier, S., Schönherr, M.: Evaluating Integration Architectures – A Scenario-Based Evaluation of Integration Technologies. In: Draheim, D., Weber, G. (eds.) TEAA 2005. LNCS, vol. 3888, pp. 2–14. Springer, Heidelberg (2006)
2. Berg, M.: Health Information Management: Integrating Information Technology in Health Care Work. Routledge (2004)
3. Isern, D., Sanchez, D., Moreno, A.: Agents applied in health care: A review. International Journal of Medical Informatics 79(3), 145–166 (2010)
4. Kirsh, W. (ed.): Encyclopedia of Public Health, vol. 1. Springer Science (2008)
5. Miranda, M., Pontes, G., Gonçalves, P., Peixoto, H., Santos, M., Abelha, A., Machado, J.: Modelling Intelligent Behaviours in Multi-agent Based HL7 Services. In: Lee, R. (ed.) Computer and Information Science 2010. SCI, vol. 317, pp. 95–106. Springer, Heidelberg (2010)

An Architectural Pattern for Designing Intelligent Enterprise Systems

Eugenio Zimeo, Gianfranco Oliva, Fabio Baldi, and Alfonso Caracciolo

Abstract. Social mining, recommenders and data semantics are moving the focus of enterprise systems towards context-awareness and personalization. However, the design of these software systems needs specific architectures to support intelligent behaviors, still ensuring important non-functional properties, such as flexibility, efficiency and scalability. This paper proposes an architectural pattern that helps designers to easily identify the subsystems that characterize intelligent enterprise systems. By decoupling transactional behavior from batch processing, the pattern avoids the interference of knowledge extraction and reasoning processes with the state and the performance of the transactional subsystem. The pattern has been experimented in e-Commerce by designing an intelligent and scalable virtual mall.

Keywords: Architectural Pattern, Software Systems Design, Enterprise Systems, Intelligent Systems, e-Commerce.

1 Introduction

Enterprise systems represent today an important class of large-scale software that supports many fundamental processes of complex organizations, ranging from resource planning to business intelligence. In this class, e-Commerce applications often groups many enterprise assets to offer an integrated and coherent view to merchants and customers for performing business actions.

In the evolution of the Web, Web 2.0 technology represents a milestone with its emphasis on supporting social collaboration and reasoning over data semantics. The

Eugenio Zimeo
University of Sannio, Dep. of Engineering, 82100 Benevento, Italy
e-mail: eugenio.zimeo@unisannio.it

Gianfranco Oliva · Fabio Baldi · Alfonso Caracciolo
Poste Italiane S.p.A., TI-RS-Centro di Ricerca e Sviluppo, 80133 Napoli, Italy
e-mail: {olivag11,baldifa3,caracc52}@posteitaliane.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 89–96.
springerlink.com

new way of interaction is introducing a radical innovation in e-Commerce [18], moving the focus towards context-awareness and personalization. If on the one hand, these innovative features improve product selling, they also significantly impact the way modern e-Commerce applications are designed and implemented.

Traditional architectures exploited to design enterprise systems (in particular e-Commerce applications) need to be revised in order to take into account the desired ability of these systems to "generate" knowledge while working, on the basis of several information sources that could be available as enterprise assets. These sources could be tightly related to the e-Commerce application or belonging to different enterprise subsystems that support cross-business features; they can change over time or can be enriched with additional ones when new needs emerge.

Designing an architecture for e-Commerce applications in this new scenario is not simple, since designers have to combine the expected intelligent behavior of the system with non-functional requirements, such as flexibility, efficiency and scalability. Separation of concerns, already applied to design complex architectures by separating orthogonal non-functional aspects in insulated modules, could be a viable approach also to design flexible and scalable intelligent enterprise systems.

This paper presents an architectural pattern that helps satisfying both functional and non-functional scalability during the design of intelligent enterprise systems. The pattern decouples the transactional behavior from batch processing, in order to avoid dangerous interference of knowledge extraction and reasoning processes with the state and the performance of the transactional subsystem. The pattern is applied to the design of a complex e-Commerce application, which virtualizes a *mall* composed of a variable number of eShops. The pattern is adopted as a key step of the ADD - *Attribute Driven Design* method [15], which is exploited as a reference process model to design the whole system. The paper shows the effects of the pattern to simplify the comprehension of the system and to improve its design.

The rest of the paper is organized as follows. Section 2 discusses some common architectural patterns exploited to design e-Commerce applications and the role of patterns in designing intelligent systems. Section 3 presents the *Inference Pattern* proposed in this paper. Section 4 briefly discusses the application of the pattern to design a real system. Finally, Section 5 concludes the paper.

2 Background and Related Work

Architectural patterns, design patterns and idioms constitute an extensible knowledge base that helps designers to take proper decisions in short times, when they design complex software systems, by reusing solutions already experimented in similar application contexts.

Several architectural patterns have been proposed in the literature for different classes of software systems. An interesting proposal comes from IBM [1, 2]. It identifies several patterns at different abstraction levels to guide the design of e-Business applications, a specific class of enterprise systems including e-Commerce. In this vision, user requirements drive the selection of one or more Business patterns among

Self-Service, Collaboration, Information Aggregation, and Extended Enterprise. These are used to suggest the adoption of *application* and *integration* patterns, often called *architectural* patterns. The formers capture the functional and non-functional requirements of an application, by proposing the system decomposition in functional and logical subsystems. The latters suggest a way to integrate different subsystems: *Access Integration* patterns define a common entry point for accessing services whereas *Application Integration* patterns represent a way to integrate the flow of actions or the data owned by different subsystems. In the literature, a lot of integration patterns, mainly based on messaging, has been introduced to support Enterprise Application Integration (see [3] for a presentation of these patterns).

A widespread architectural pattern that satisfies self-service interaction model is *Model View Controller* (MVC) [4, 5]. This pattern, with its main variants (*Model View Presenter* [4] and *Presentation Abstraction Controller* [7] are the most known), suggests the organization of the presentation layer of enterprise systems. However, additional patterns are needed to help the design of the other layers proposed by multi-layer and multi-tier decompositions [8].

Service Oriented Architecture (SOA) is becoming the reference high-level architectural pattern to design flexible, reusable and dynamic enterprise and inter-enterprise systems, especially when asynchronous interactions, based on an *Enterprise Service Bus*, are exploited. In [9], the authors propose a mediator-based architecture that decouples service providers from consumers with the aim of binding services on demand on the basis of customers' preferences.

However, the "intelligent" dimension of enterprise systems needs more autonomous and proactive behaviors to satisfy users' needs. Autonomic computing [10] is emerging as a promising approach to include self-* properties in software design and workflow systems [19], whilst MAPE-K (Monitor, Analyze, Plan, Execute and Knowledge) [11] is often used as a reference architectural pattern to design these software systems. This pattern allows for observing the state of a resource in order to intercept possible deviations from a desired behavior that can be controlled by planning adaptations. However, it represents only a starting point for intelligent enterprise systems, where specific patterns are needed to design systems with the ability of inferring knowledge from data generated during the execution.

The paper in [17] focuses on a knowledge management architecture to apply data mining to e-Commerce but it does not address the general architecture of the system. In [12], the authors discuss an architecture to efficiently perform OLAP on the data produced by customers through the interaction with an e-Commerce site. Even if, the idea of closing the knowledge loop - users, application, analysis, application changes, users - is similar to the one proposed in this paper, the architecture does not provide users with sufficient hints about the identification of finer-grained subsystems.

In this paper, we present a new architectural pattern, called *Inference Pattern*, that provides software architects with a conceptual framework for designing the business logic of intelligent enterprise systems.

3 Inference Pattern

The pattern will be described according to a typical schema for patterns.

Problem. We want to design a software system able to perform processing on its own data, produced by user interactions, with the aim of expliciting the implicit knowledge that may be useful to users and to the system itself.

Context. Interactive applications generate data from users interactions, which are compliant to models that are typically described through relational, ontological or object-oriented schemas. Data passed during interactions can be processed with the support of data coming from other sources to generate new information that can be useful to understand users' behaviors and preferences. This knowledge, often defined implicit or tacit, can be exploited to personalize the interaction between users and the system, since it eases the knowledge transfer from software to users, by reducing the effort to access to data. This improves the effectiveness of the interaction with reference to the business objectives of the system. The knowledge acquired by the system can be in turn reused by other systems.

Solution. The system is decomposed in two logically separated subsystems: the one in charge of answering to user requests and the one responsible of collecting data and preprocessing them. The former becomes passive with reference to the latter that, on the other hand, generates the aggregated data that are useful to improve user interactions and the quality of the data hosted by the interactive subsystem. The logical separation promotes both the knowledge base extensibility, thank to the possibility of using additional systems as subject, and the continuous and concurrent processing with respect to the transactional activity generated by the interactive subsystem, so easing and optimizing the successive deployment.

Participants. In the following, a list of participating subsystems (see Fig. 1) is given with a brief description.

- Subject: is the observed subsystem from which implicit knowledge is extracted.
- *Knowledge extraction:* collects the data produced by the interactions between users and the subject and extracts the implicit knowledge.
- *Generation:* generates new knowledge for the subject, by using the implicit knowledge extracted by the Knowledge Extraction (KE) subsystems.
- *Decision Support:* it provides the user with a new knowledge extracted by the KE subsystem to support proper decisions.
- *Recommendation:* suggests information to users. It may use the Generation subsystem to ask for the generation of aggregated data or the Decision Support subsystem to recommend a choice to a user.
- *Personalization:* it personalizes the suggestions with respect to the specific user (or context in general). It is a sort of recommender supporting personalization.

Structural view. In Fig. 1, the structural view of the pattern is described by highlighting the participants introduced above and the *use* relationships among them. The red dotted line is a particular use dependency since it, differently from the other ones, works on the Subject by writing onto it. In addition, the view shows two subsystems

of KE: Management and Search, which can be useful to configure the inference rules and to perform particular search operations on the collected data.

The KE subsystem can access to the Subject by means of synchronous (dependency regards the external operations of the subject) or asynchronous (dependency is related to the events) interactions.

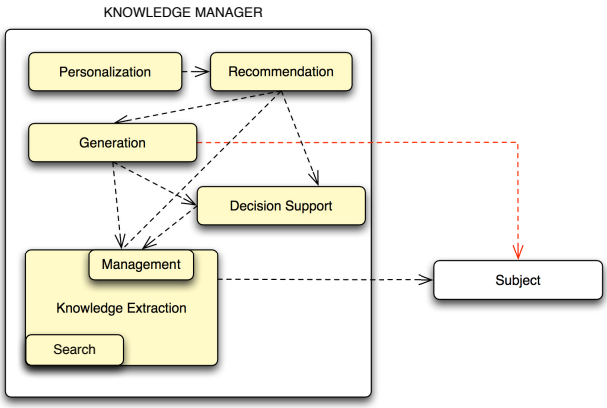


Fig. 1 Structural view of the pattern

Related patterns. The proposed pattern is inspired by two known patterns: *Observer* [4] and *MAPE*. The former is a design behavioral pattern that introduces the concept of observing the events generated by a Subject (or Observable system) to drive the behavior of a multitude of Observers. The latter, on the other hand, changes the viewpoint with respect to the subject, which becomes a resource monitored by a manager that is able to analyze the state of the resource and to apply to it, if needed, some state changes (that in turn may produce behavioral changes).

The Inference Pattern, differently from MAPE, does not aim at conferring an autonomic behavior to the system but mainly at providing users with an additional knowledge inferred from the data collected from the subject. Therefore, even if, like MAPE, it creates a closed-loop system, the closure is mainly performed by users and, when possible, by the system itself through the writing operations (Generation) applied to the state of the system. Typically, these operations do not change the behavior of the system, but they make it possible to retrieve further information by users, to feed the knowledge cycle.

Example. Fig. 2 shows an example of pattern application in a typical case that combines interactive transactions derived from users (manual operations that store data coming from users' knowledge) and concurrent data extraction and elaborations for (automatic) recommendations to the same user. The (software) agent is in charge of performing the use cases that regard automatic operations. The results of these operations represent the inferred knowledge that is used to provide suggestions in a given context. The figure also shows the separation of the subsystems in

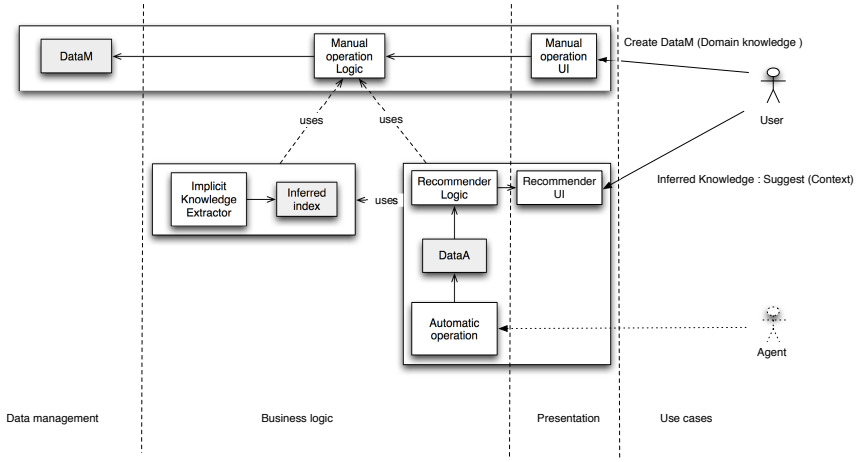


Fig. 2 Generic case of pattern application

finer grained logical ones, useful to implement a layered architecture that can be easily transformed in a multi-tier one thanks to the separation of the subsystems onto different resources.

4 Pattern Application: The InViMall System

The Inference Pattern has been exploited to design a scalable e-Commerce system, called InViMall (*In*telligent *Vi*rtual *Mall*), at Poste Italiane S.p.A. InViMall is an electronic mall, based on e-community concepts, that enhances the shopping experience of users. It exposes several innovative features through a multichannel interface, taking into account typical non-functional requirements of enterprise systems: *Personalized suggestions* [13], *Automatic generation of personalized product bundles* [14], *Advanced marketing intelligence*, *Faceted Navigation*.

The design of InViMall was performed by following the *Attribute Driven Design* method proposed in [6]. According to this method, it is important to clearly define the requirements, both functional and non-functional with the aim of identifying the architectural drivers that can satisfy the requirements. Therefore, the first phase of the process was the identification of scenarios and use cases that cover the innovative aspects of the system. Hence, some functional areas and possible subsystems were initially identified.

Due to the absence of a reference architectural pattern for the business logic of this kind of system, the next step was the identification of dependencies among subsystems, by using use cases analysis, with the aim of building a *Dependency Structure Matrix* (DSM) [16]. It was useful for identifying and analyzing the mutual dependencies among the subsystems.

Reasoning only about the functional areas created a dependency graph with several mutual-dependencies, so generating a lot of coupling in the systems that could negatively impact flexibility, reusability, performance and scalability.

To overcome the problem, the architectural drivers of the InViMall system were generalized and an effort was done to abstract a solution by defining the architectural pattern, presented in the paper. The application of the pattern caused a complete refactoring of the design, introducing a strong reduction of mutual dependencies and a clear separation between the interactive and transactional subsystem (called *Mall*), and the batch subsystem (called *Intelligent Mall*). The former groups the typical subsystems that characterize an electronic mall, extended with social network features. The latter hosts (i) a KE, which implements knowledge reasoning, predicted rating features and social mining, (ii) a decision support subsystem that enables merchant to configure and manage innovative marketing campaigns, (iii) a *Generation* subsystem that works onto the state of the Mall to store new explicit knowledge extracted by KE from the implicit one collected from several sources. It is worth to note that several features were cross-cutted among different subsystems before applying the pattern, so generating mutual dependencies and coupling.

The two main subsystems derived by the decomposition have different characteristics that impact their implementation. The Subject is a typical transactional system and therefore it is characterized by concurrent accesses, data persistence and ACID transaction management. On the contrary, the Knowledge Manager is a batch system using background processes to manipulate large amounts of data for analytical processing and to generate correlations among entities.

To avoid bottlenecks at control layer that may reduce concurrency and improve coupling, an event-driven layer is suggested to handle the events coming from the user interface and to propagate them to the specific subsystems.

5 Conclusion

The paper presented an architectural pattern that helps software architects to design intelligent enterprise systems that combine interactive features with batch processing to infer knowledge from large amounts of data. The pattern was applied with success to design a complex application that implements an electronic mall. Thanks to it, software architects clearly and rapidly identified the functional subsystems, by associating to them use cases. The pattern, also, contributed to create a loosely coupled system with a clear logical separation between the transactional and intelligent subsystems, and a further separation among the subsystems of the intelligent side.

The pattern enables the integration of both real-time, event-driven interactions, and data accesses for large volume of data. In the future, more sophisticated integration middleware platforms that transparently combine the two benefits above will be investigated.

Acknowledgements. This work was partially supported by MSE under the Intelligent Virtual Mall (InViMall) Project MI01-00123.

References

1. http://www.opengroup.org/togaf/
2. http://www.ibm.com/developerworks/patterns/
3. Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison Wesley (2003)
4. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley (1995)
5. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture - A System of Patterns. John Wiley & Sons (1996)
6. Potel, M.M.: Model-View-Presenter: The Taligent Programming Model for C++ and Java (1996)
7. Plakalovic, D., Simic, D.: Applying MVC and PAC patterns in mobile applications. Journal of Computing 2(1) (2012)
8. Fowler, M.J.: Patterns of Enterprise Application Architecture. Addison Wesley (2002)
9. Cong, S., Hunt, E., Dittrich, K.R.: IEIP: An Inter-Enterprise Integration Platform for e-Commerce Based on Web Service Mediation. In: ECOWS, pp. 201–210 (2006)
10. Kephart, J., Chess, D.: The vision of autonomic computing. Computer 36(1), 41–50 (2003)
11. Huebscher, M.C., McCann, J.A.: A survey of autonomic computing - degrees, models, and applications. ACM Comput. Surv. 40(3), 1–28 (2008)
12. Ansari, S., Kohavi, R., Mason, L., Zheng, Z.: Integrating E-Commerce and Data Mining: Architecture and Challenges. In: International Conference on Data Mining, pp. 27–34. IEEE (2001)
13. Birtolo, C., Ronca, D., Armenise, R.: Improving accuracy of recommendation system by means of Item-based Fuzzy Clustering Collaborative Filtering. In: 11th International Conference on Intelligent Systems Design and Applications - ISDA 2011. IEEE, Spain (2011)
14. Birtolo, C., De Chiara, D., Ascione, M., Armenise, R.: A Generative Approach to Product Bundling in the e-Commerce domain. In: The 3rd World Congress on Nature and Biologically Inspired Computing - NaBIC 2011, pp. 169–175. IEEE, Spain (2011)
15. Bachmann, F., Bass, L.: Introduction to the Attribute Driven Design Method. In: The 23rd International Conference on Software Engineering - ICSE (2001)
16. Clements, P., et al.: Documenting Software Architectures: Views and Beyond. 2nd edn. Paerson Education (2011)
17. Yu, H., et al.: Knowledge Management in E-commerce: A Data Mining Perspective. In: Management of e-Commerce and e-Management. IEEE (2009)
18. Guo, S., Wang, M., Leskovec, J.: The Role of Social Networks in Online Shopping: Information Passing, Price of Trust, and Consumer Choice. In: Conference on Electronic Commerce. ACM (2011)
19. Polese, M., Tretola, G., Zimeo, E.: Self-adaptive management of Web processes. In: Web Systems Evolution (WSE), pp. 33–42. IEEE (2010)

MUSE: MUltilinguality and SEmantics for the Citizens of the World

Michele Bozzano, Daniela Briola, Diego Leone, Angela Locoro, Lanfranco Marasso, and Viviana Mascardi

Abstract. This paper discusses some of the challenges raised by multilinguality in the Public Administration field and shows how the MUSE system addresses them by combining speech to text, text to speech, and machine translation techniques, utilizing domain ontologies throughout a complex system designed as a Multi-Agent System and deployed by exploiting resources in the Cloud. The design of MUSE is finished and its implementation and unit testing are under way. On completion of these two stages, MUSE will be experimented in the Registry Office of Genoa Municipality which is supporting this activity by providing the authors with data, advice on the domain, and the opportunity to test MUSE on the field. The final purpose of this work is to make MUSE's services available to all the foreign citizens interacting with Genoa Municipality's Registry Office. The two languages currently supported are Spanish and Italian. Due to the high modularity of MUSE's architecture, a limited effort will be required to add other languages in the future, if the on site experimentation will be successful.

1 Introduction and Motivation

In a world that is becoming smaller and smaller thanks to increasingly efficient and cheap transportation facilities, many persons move for many reasons - tourism, education, migration - and whatever the reason, they are often faced with difficulties because of the lack of understanding of the hosting country's language.

Michele Bozzano · Daniela Briola · Angela Locoro · Viviana Mascardi
Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS), University of Genova, Italy
e-mail: {michele.bozzano,daniela.briola,angela.locoro, viviana.mascardi}@gmail.com

Diego Leone · Lanfranco Marasso
Engineering Ingegneria Informatica, Genova, Italy
e-mail: {diego.leone,lanfranco.marasso}@eng.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 97–102.
springerlink.com

The MUSE system, whose name originates from "MUltilinguality and SEmantics for the Citizens of the World", aims at supporting a foreign citizen in her attempt to interact with the Public Administration (PA) of the hosting country in her own language.

The user is driven by the system to go straight to the point and do the right moves to achieve her goals towards a happy end transaction with the PA. The user friendly interface has been designed to let the user elaborate her requests in natural language, utter a question and receive a spoken answer in her native language. The answer describes the necessary steps that have to be taken from a bureaucratic point of view (documentations or other offices where to address) in order to complete the life-cycle of the user's request. The user can be asked to reformulate her request in case her question is not clear or is considered out of context by the system. If the question implies that more paths of interaction are possible, and the system cannot decide autonomously which one is correct, multiple choices are proposed to the user for a further refinement. MUSE is able to improve its knowledge of the common requests that can be formulated by users, as it evolves and learns what are the more frequent queries. It also collects all the interactions in order to extract similar patterns of behavior and enriches it for optimization purposes.

MUSE, whose design is completed and whose implementation is under way, will be experimented in the Registry Office of Genoa Municipality.

The paper is organized in the following way: Section 2 presents the architectural, functional, and implementation details of the system, and Section 3 concludes.

2 The MUSE System

MUSE is based on speech translation technologies [1], machine translation [2], ontologies [3], intelligent agents and multi-agent systems (MASs) [4], and cloud computing [5]. The purpose of our work is to make MUSE's services available to all the foreign citizens interacting with Genoa Municipality's Registry Office first, and to other PAs in the future. Spanish and Italian are the languages currently supported inside the system.

As far as the architectural and technological choices behind MUSE are concerned, we decided to integrate some modules as "components out of the shelf" and to treat them as black box items. For example, the Spanish-Italian and Italian-Spanish translations result from the exploitation of the Google Translate service API[1]. This is also done for modularization purposes. Once a change in MUSE is decided, i.e. it has to manage more than one language, it should be sufficient to change the language pair translation in the API, while the formalized part of the system, namely the ontology and the procedural rules, either remains unchanged or just needs to undergo a conservative extension.

Once the user query is uttered in Spanish, transformed into a Spanish text, and translated into the corresponding text in Italian, a query expansion procedure is run to help disambiguate the request and find a match with one of the "well known

[1] http://research.google.com/university/translate/index.html

problems" encoded in the MUSE ontology. The query expansion is carried out by means of a translation table, that contains keywords and phrases patterns, each associated with one of their interpretations in terms of ontology concepts.

The sub-domain of practices related with documents proving the identity of a citizen is used in the sequel for better explaining our approach. The ontology which codifies the different "well known problems" that correspond to user requests has been created in Italian which we adopted as "lingua franca" inside the system. Figure 1 depicts a portion of the OWL ontology created with Protégé[2], that results from such codification. The meaning of concepts from the topmost one, and following a left-to-right BFS visit, are: "Identity Card"; "Identity Card of an Adult"; "Identity Card of a Minor"; "Identity Card First Issue"; "Identity Card Renewal"; "Renewal for Personal Data Change"; "Renewal for Address Change"; "Renewal for Expiry"; "Renewal for Loss or Theft"; "Renewal for Deterioration". This is a portion of the actual ontology used in MUSE, and the choice of Italian for representing its concept was due to the need of easing our interaction with the staff from Genoa Municipality, and of providing them with a formalization of their domain that they could feel comfortable with and that could even manage by themselves in the future (providing a suitable user interface). Similar considerations hold for the procedural rules discusses later on in this section, and encoded using strings in Italian.

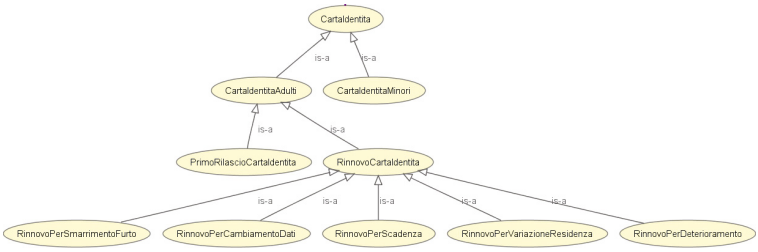


Fig. 1 MUSE ontology for "identity well known problems"

The aim of the ontology is to drive the system to retrieve the correct procedural rule against the user query, once the query has been properly expanded and interpreted. To this aim, the ontology contains all the different paths elicited by the domain experts for the identity card sub-domain. The intermediate concepts are the names of the problems, which have been refined at different levels of depth. The leaves of the ontology represent the head of the procedural rule that codifies the procedure to be activated, once assessed that it corresponds to the procedure associated with the user's request. Such procedural rules are represented as plans in the Jason declarative agent-oriented programming language [6] which is seamlessly

[2] http://protege.stanford.edu/

integrated into JADE[3]. Jason turned out to be a very convenient choice both for *representing procedural rules in a declarative, compact, and not ambiguous way*, and for *executing them*. Jason plans corresponding to procedural rules consist of:

- the "triggering event" that must match a leaf concept in the ontology;
- an optional "context" stating under which conditions the plan can be applied;
- the "body" consisting of a sequence of steps necessary to complete the procedure.

Strings starting with "//" are comments.

For example, in case the user formulates a query for a change of address in her identity card, the triggering event of the rule, according to the ontology, will be *"rinnovoPerVariazioneResidenza"* - renewal for address change. The Jason plan that represents the correct procedural rule for dealing with this problem (topmost one in Figure 2) has no context, meaning that is can be always applied, and the steps codified into the rule are *"portareTreFotoFormatoPassaporto"* - provide three passport-sized photos - and *"portareDocumentoIdentitaValidoRecente"* - provide your more recent valid identity card.

Each step is further extended into a set of Jason rules whose triggering event is the step itself, and the context (the atom between the semicolon and the arrow) states which language should be used as the output of the text to speech process. The information of the current language is set at the beginning of the conversation.

Actions in the body of the last four rules shown in Figure 2 are of type ".say": the system that executes the rule must provide a text to speech service and call it when the ".say" action is executed within a body's rule. Actions might be of any kind, including access to information systems and other applications.

Rules in MUSE are codified by hand based on documents produced by Genoa Municipality's Registry Office and on interviews to the office staff.

One of the most challenging activities in the design of multilingual systems is how and where to collect native language utterances and queries that users do while interacting with the tool. For this reason a questionnaire has been prompted and submitted to users in form of short interviews that were conducted at the end of a typical interaction with a human operator, and interactions were recorded (asking the users to sign a module to state that they agree with the recording, and respecting all the norms regulating privacy issues). Both the recorded interactions and the questionnaire are exploited to identify some useful dimensions of the dialog between humans that can drive the system design towards an improvements of such aspects. The questions posed to the user at the end of a session are the following (expressed in the user's native language):

- Which question did you make to the operator?
- Did you obtain the answer you were looking for?

[3] JADE (Java Agent DEvelopment Framework, [7]) is a software framework implemented in Java which provides a middleware and a set of graphical tools that support communication, debugging and deployment of MASs. The agent platform can be transparently distributed across machines. Jason is a platform for the development of MASs developed in Java and supporting different infrastructures for the deployment of MAS including JADE.

```
+!rinnovoPerVariazioneResidenza <- // triggering event
   !portareTreFotoFormatoPassaporto; // step 1 of the rule's body
   !portareDocumentoIdentitaValidoRecente. // step 2

+!portareTreFotoFormatoPassaporto : spagnolo // context ES
   <- .say(es, "Debe proporcionar tres fotos tamano pasaporte").

+!portareTreFotoFormatoPassaporto : italiano // context IT
   <- .say(it, "Portare tre foto formato passaporto").

+!portareDocumentoIdentitaValidoRecente : spagnolo
   <- .say(es, "Debe proporcionar el ultimo documento de identidad").

+!portareDocumentoIdentitaValidoRecente : italiano
   <- .say(it, "Portare un documento d'identita valido recente").
```

Fig. 2 Jason plans for identity card renewal

- How difficult was understanding what you have to do?
- How difficult are the steps that you are now expected to face in order to complete the procedure?
- Was the time you were engaged into the conversation too short, too long or just fitting your expectations?

Some users may do mistakes when formulating their queries. MUSE records such mistakes and the relative correction, and ranks the pairs obtained as they are repeated during real time interactions. In this way the system increases the strength of its hypothesis on the occurrence of such patterns more often than chance, and may provide an automatic correction when the same situation happens again. The users themselves offer feedback when they select their intended request against a set of equally possible requests that the system may show them, hence modifying the ranking of translations based on such feedback for similar queries. In this way the system is able to show as top ranks the most likely translations associated with the most likely requests.

The system, developed in JADE, will be distributed across the cloud thanks to APIs that "Engineering Ingegneria Informatica" has developed for the Java language in collaboration with the DIBRIS Department of Genoa University [8].

3 Conclusions

In this paper we described the MUSE system whose implementation is almost completed, and that will be experimented in the Registry Office of Genoa Municipality.

The foreseen advantages in using MUSE for the Registry Office are that misunderstandings and bad translations troubles should be minimized, as well as the user sense of uneasiness; the difficulties in understanding the answers should disappear, as they will be provided in native language; the queues at the registry office should

be shortened, as the system might contribute to address the users to the right place; as a consequence, latency time might be reduced for Italian citizens as well.

Even if MUSE will be first experimented on the specific Registry Office domain and using Spanish and Italian only, its modular and distributed architecture makes it suitable to provide intelligent speech to speech services, guided by a deep knowledge of the domain, in any scenario and involving any language. Besides adapting MUSE to other PAs, we plan to tackle the "Indiana MAS and the Digital Preservation of Rock Carvings" scenario, in order to allow archaeologists from different countries to interact and share opinions in their own language about interpretation and dating of rock art artifacts. We have already defined some ontologies describing Mount Bego's rock engravings and we have access to a large amount of data and documents [9]. Supporting multilinguality is one of the most challenging objectives of Indiana MAS, and we are confident that MUSE will be the right application for coping with it.

Acknowledgements. This work is partially supported by the "Indiana MAS and the Digital Preservation of Rock Carvings" MIUR FIRB Project funded by the Italian Ministry of Education, Universities and Research.

References

1. Wahlster, W. (ed.): Vermobil: Foundations of Speech-to-Speech Translations. Springer (2000)
2. Hutchins, W.J., Somers, H.L. (eds.): An introduction to machine translation. Academic Press (1992)
3. Gruber, T.R.: Knowledge Acquisition 5, 199 (1993)
4. Jennings, N.R., Sycara, K.P., Wooldridge, M.: Autonomous Agents and Multi-Agent Systems 1(1), 7 (1998)
5. Mell, P., Grance, T.: The NIST definition of Cloud Computing. Tech. Rep. 800-145, National Institute of Standards and Technology (2011)
6. Bordini, R.H., Hubner, J.F., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons (2007)
7. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. John Wiley & Sons (2007)
8. Ghio, S.: Progetto di un prototipo di SDP per utilizzo in Pubblica Amministrazione e reti private. Master's thesis, Computer Science Department. Genova University, Italy (2012) (in Italian)
9. Papaleo, L., Quercini, G., Mascardi, V., Ancona, M., Traverso, A., Lumley, H.D.: In: ICAART 2011 Procs (2011)

Ontology Based Road Traffic Management

A.J. Bermejo, J. Villadangos, J.J. Astrain, and A. Córdoba

Abstract. Every time an abnormal situation happens on the road, danger situations are found and they need the skills of the drivers in order to cope with them. In this paper, we propose to integrate an ontology inside each vehicle to provide them with reasoning capabilities and avoid the use of a central decision point. Our system benefits traffic management in emergency situations with immediate decisions where each vehicle is a decision point, reacts considering its neighbouring vehicles, and collaborates with them to reach consensus in real time.

Keywords: VANET, traffic management, decision, ontology, emergency.

1 Introduction

Nowadays, safety on the roads is the main goal that Governments and companies are pursuing. It includes situations in which an emergency vehicle needs to have free way in order to carry out its job as fast as possible.

The aim is to manage the traffic distribution, vehicle by vehicle individually, in order to make the emergency vehicles having free way in the minimum time possible and avoiding risk situations on the road.

Every time an abnormal situation happens on the road, danger situations are found and they need the skills of the drivers in order to cope with them. These special situations can happen because of different reasons. Some of them could be: an object on the road, weather conditions, the scenario or an incoming emergency vehicle on the road.

On the one hand, in most situations the drivers act correctly but not in the best way, and it could be much better. On the other hand, in plenty of situations the reactions of the drivers are completely incorrect and could cause (actually, sometimes

A.J. Bermejo · J. Villadangos · J.J. Astrain · A. Córdoba
Dpt. Ingeniería Matemática e Informática, Universidad Pública de Navarra,
Campus de Arrosadía, 31006 Pamplona (Spain)
e-mail: {antoniojesus.bermejo,jesusv,josej.astrain,
alberto.cordoba}@unavarra.es

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 103–108.
springerlink.com

they cause) new danger situations and even accidents. The current scenario is particularly complicated because there is already an emergency situation which is trying to be solved by the emergency vehicle, and new situations of this type could be generated. It would mean different drawbacks to study:

- The expected service provided by the emergency vehicle would be retarded and, consequently, the life of other people could be endangered.
- There would be new danger situations, which would affect new users and the current users of the road. They would have to take a special care in their behaviour while the situation is happening.
- New accidents could easily happen, becoming the situation even more drastic and dangerous.

Therefore, there is a necessity of studying this scenario and proposing a range of ideas which could help in this type of situations with uncertainty, in order to happen less often or even disappear.

Contributions

We consider, as other proposals do, that vehicles are equipped with sensors and communication devices that collect and communicate sensor data. In our case, emergency situations are handled in a different way: we integrate an ontology inside each vehicle in order to provide them with reasoning capabilities and to avoid the use of a central point.

Our system takes advantage of the information collected in real time, allowing immediate, intelligent and distributed decisions in order to better manage the traffic when emergency situations occur. Each vehicle, as a decision point it is, reacts considering the scenario and the information provided by its neighbouring vehicles, and collaborates with them to reach consensus in a very short time period.

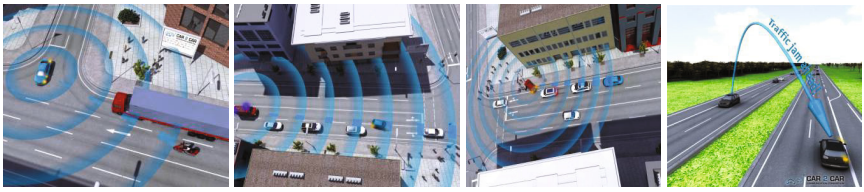
Considering the case of highways, the strategy used to design the ontology is to clean the fastest lane by trying to move all the other vehicles to the lowest lanes. Thus, the emergency vehicle will have free way and will be able to give its service in a minimum time.

2 Related Work

There are many projects working on this topic. Some of them are: Car2Car-CC [2], SimTD [7], GST Rescue [4], PReVENT, FleetNet [1], DSRC, SeVeCom [12], NoW [5], Coopers, Anemone and Poseidon. All of them are related with Vehicular Ad-hoc Networks (commonly called VANETs).

The most relevant emergency scenarios are shown and explained in [2], but without providing any solution. The different scenarios, depicted in Figure 1, are: a) Motorcycle warning (view obstruction); b) approaching emergency vehicle; c) warning of roads work (view obstruction); and d) avoidance of traffic jams. Most of the proposals in the literature focus on studying the situation of the roads, obtaining

information about traffic congestion [8] and giving alternative routes [10] to the emergency vehicles if there are traffic jams or slow traffic flow. In most cases information is collected at a central point to take decisions and vehicles are later informed.



(a) (b) (c) (d)

Fig. 1 Use cases (Car2Car Communication Consortium)

Another approach is presented in [3], where emergency vehicles inform to a central point, which disseminates some basic orders given to the vehicles in the way of the emergency vehicle. In this case, the reasoning capacity is in the central point. Vehicles are notified about the situation and the actions to perform before the arrival of the emergency vehicle to their position, in order to give them enough time to react properly and without any risk. Traffic signals and other concrete devices on the road, under the control of the central point, are gathering/providing updated information from/to the vehicles on the road. And traffic signals are configured by the intelligent central point to manage the traffic in such emergency situations.

There have been some basic tests done in real situations [3, 9], and the results have been quite successful. In [9], a concrete scenario is analysed: a straight road with three lanes in the same direction, and only three possible lane changes for a vehicle: no change; overtakes the slower ones; gives pass to the faster ones. Table 1 summarizes both requirements and solutions provided by current proposals.

Table 1 General comparison with other works

	Requirements	**Current Proposals**
Problem Studied	Managing every single vehicle in emergency situations	Avoiding traffic congestion and studying dangerous scenarios
Area Concerned	Small area, normally not more than one road or street	Wide areas with many streets and roads
Solutions Provided	Detailed solutions for every single vehicle on the road	No solutions provided or too simple and general ones

Current proposals consider that vehicles are equipped with sensors and communication devices that collect and communicate sensor data. However, vehicles do not integrate reasoning capabilities, which is left to a central point. Traffic management in emergency situation could benefit from immediate decisions avoiding the use of

a central point. In such case, each vehicle collects its perception of the scenario and aggregates the situation awareness provided by other vehicles located close to it. The use of a common ontology ensures a quick and effective response to the event. As all the vehicles share the same ontology, all of them reach a consensus and act together avoiding inconsistent decisions.

3 Ontology-Driven Decision System

Vehicles ontology, based on sensor data, suggests driver different actions depending on the situation. The ontology extends the A3ME [6] ontology by adding some classes to model each vehicle respect its neighbours: position, distance and acceleration. We extend the class 'Output' to model ontology suggestions: Accelerate, SoftAccelerate, Decelerate, SoftDecelerate, MaintainSpeed, MaintainDistanceWithCarInFront and ChangeLane.

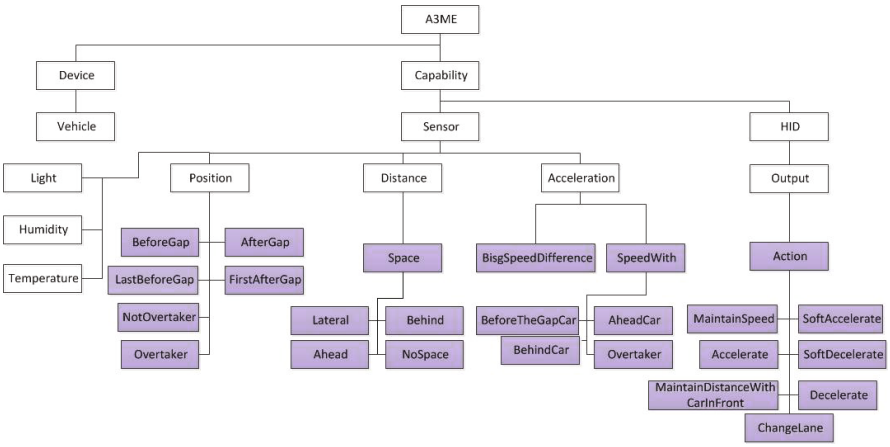


Fig. 2 Proposed ontology extended from A3ME ontology

In Figure 2, purple rectangles represent the classes we have added to the A3ME ontology. In our case the ontology has 24 classes, 12 properties, and 77 rules. Additionally, there are integrated other classes that we will use to adapt decisions to weather conditions.

The complete process followed since the gathering of information until the concrete vehicle performing an action is broken down. This process is exactly the same for every single vehicle on the road.

1. There is a device installed inside the vehicle that is able to exchange information.
2. When there is an emergency situation on the road, vehicles broadcast an alarm towards all the other vehicles on the road.
3. Each vehicle, by reception of the alarm signal, uses the semantic included in it in order to extract agreed suggestions.

4. Suggestions are provided to the driver through a screen located inside the vehicle, a voice system or both.
5. The driver reacts taking into account the suggestions. Here is the common sense of the driver which is the main factor.

4 Scenario Evaluation

Figure 3 presents two emergency scenarios, and also the actions suggested by the ontology. We have evaluated these scenarios with different number of vehicles and measured the time required to extract the suggestions. We have considered from five to nine vehicles distributed uniformly among the lanes, because decisions are taken for each vehicle and its neighbours (5 vehicles in line occupy about 50 meters).

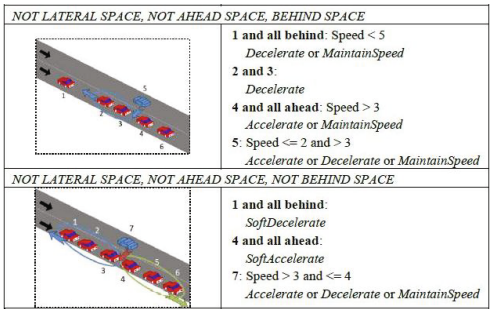


Fig. 3 Emergency scenarios, and corresponding ontologyś suggestions. Vehicles are numbered from 1 to 7 (left side); and react as suggested by the ontology (right side).

As the system is dynamic, every movement is detected and shared instantaneously. Thus, if a driver acts in a different way than expected, it will be detected and there will be adapted suggestions to it. Table 2 shows that the inference process is around 300 ms. It is showed that our system could operate in real time.

Table 2 Performance evaluation of system

Vehicles (Overtakers)	#Total inferences	Time/Inference (ms)
6(1)	1	343
6(2)	3	280
7(2)	3	280
9(4)	8	242

We have run different simulations with vehicles distributed into two lanes. The ones on the highest lane are the overtakers, which have to find a free space to occupy on the other lane. They are a subset of the total number of vehicles on the road.

The reduction on the time inference is due to the use of the Jess reasoner [11]. As the system progresses, the decision graph is evaluated faster.

5 Conclusions

In this paper, emergency situations are handled in a different way: we propose to integrate a ontology inside each vehicle to provide them with reasoning capabilities and avoid the use of a central point.

Our system benefits traffic management in emergency situations with immediate decisions where each vehicle is a decision point, reacts considering its neighbouring vehicles, and collaborates with them to reach consensus in real time. Our test shows inference time lower than 300 ms.

Acknowledgements. This work has been supported by the Spanish Government: TIN2011-28347-CO1-02, TIN2011-28347-CO2-02, TIN2010-17170, IPT-370000-2010-36.

References

1. Fleetnet, http://www.fleetnet.co.uk/
2. Baldessari, R., et al.: Manifesto, http://www.car-2-car.org/fileadmin/downloads/C2C-CC_manifesto_v1.1.pdf
3. Buchenscheit, A., Schaub, F., Kargl, F., Weber, M.: A VANET-based Emergency Vehicle Warning System. In: First IEEE Vehicular Networking Conference (VNC 2009), Japan (2009)
4. E. I. Europe. Gst rescue: Improving the response time of emergency services
5. Gerken, C.: NetWorks on wheels (1998), http://www.mindspring.com/delowgerken/vehicles/ (accessed November 02, 010)
6. Herzog, A., Jacobi, D., Buchmann, A.: A3me - an agent-based middleware approach for mixed mode environments. In: Proceedings of the 2008 The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, UBICOMM 2008. IEEE Computer Society, Washington, DC (2008)
7. Huss, S.A., Jaeger, A., Stuebing, H.: simTD: Safe and Intelligent Mobility Field Test Germany; Architecture and Applications for Car-to-Car Communication. In: 48th ACM/EDAC/IEEE Design Automation Conference (DAC 2011): Workshop on Intra and Inter-Vehicle Networking: Past, Present, and Future, San Diego (2011)
8. Mohandas, B.K., Liscano, R., Yang, O.W.W.: Vehicle traffic congestion management in vehicular ad-hoc networks. In: LCN. IEEE (2009)
9. Mohimani, G.H., Ashtiani, F., Javanmard, A., Hamdi, M.: Mobility modeling, spatial traffic distribution, and probability of connectivity for sparse and dense vehicular ad hoc networks. IEEE Transactions on Vehicular Technology 58(4), 1998–2007 (2009)
10. Rizvi, S.R., Olariu, S., Weigle, M.C., Rizvi, M.E.: A novel approach to reduce traffic chaos in emergency and evacuation scenarios. In: VTC Fall. IEEE (2007)
11. Sandia National Laboratories. Jess, the rule engine for the Java platform (September 2009), http://www.jessrules.com
12. SEVECOM. Secure vehicular communication, http://www.sevecom.org/

Autonomy and Differentiation in Distributed Systems

Ichiro Satoh

Abstract. A bio-inspired approach for self-adaptive software agents on distributed systems is presented. It introduces the notion of (de)differentiation in cellular slime molds, e.g., dictyostelium discoideum, into real distributed systems. When an agent delegates a function to another agent coordinating with it, if the former has the function, this function becomes less-developed and the latter's function becomes well-developed. This paper describes the approach and its basic evaluation.

1 Introduction

A distributed system consists of loosely-coupled components running on different computers communicate and coordinate their actions by message transfer through communication networks. a hybrid palette of methods and techniques derived from classical artificial intelligence, computational intelligence, and multi-agent systems, enables distributed systems to be adaptive, autonomous, and self-organized. This paper discusses a bio-inspired approach as a novel approach for managing new generation intelligent distributed information systems.

Many researchers have explored bio-inspired approaches for distributed systems. One of the most typical self-organization approaches to distributed systems is swarm intelligence [3, 4]. Although there is no centralized control structure dictating how individual agents should behave, interactions between simple agents with static rules often lead to the emergence of intelligent global behavior. There have been many attempts to apply self-organization and autonomy into distributed systems, e.g., a myconet model for peer-to-peer network [6], and a cost-sensitive graph structure for coordinated replica placement [5]. Bigus et al. [2] proposed an agent-based toolkit for autonomic systems, where each agent had a closed-loop controller as part for

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

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 109–114.
springerlink.com

the whole hierarchy of distributed control. The toolkit was intended to customize groups of agents but not the functions of inside agents.

However, most existing approaches tend to enable coordination between components running on different computers. Although they may be useful in several applications, their adaptations are limited within coordination between components, because they do not support any adaptation inside components. Also, existing approaches focus on their target problems or applications but are not general purpose, whereas distributed systems have a general-purpose infrastructure.

2 Approach

Cellular differentiation is the process by which a less specialized cell develops or matures to possess a more distinct form and function in developmental biology. This paper introduces the notion of (de)differentiation into a distributed system as a mechanism for adapting software components, which may be running on different computers connected through a network. Like actors [1], each component, called *agent*, is autonomous and has one or more functions, called *behaviors*, which can be invoked by other agents. Our approach introduces the undertaking/delegation of behaviors in agents from other agents as a differentiation factor. Behaviors in an agent, which are delegated from other agents more frequently, are well developed, whereas other behaviors, which are delegated from other agents less frequently, in the cell are less developed. Finally, the agent only provides the former behaviors and delegates the latter behaviors to other agents.

Each agent has one or more functions with weights, where each weight corresponds to the growth of its function. When an agent wants to execute a function, if one or more agents, including itself, have the function, it selects the function whose weight is the largest among the weights of the same behaviors and delegates the selected function. When an agent delegates the execution of a function provided in another agent, the weight of the function is creased and the latter agent multicasts *restraining* messages in order to decrease the weights of the same functions provided in other agents. As a result, agents complement other agents in the sense that each agent can provide some functions to other agents and delegate other functions to other agents that can provide the functions.

3 Design and Implementation

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

3.1 Agent

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

The agent has behaviors $b_1^k, \ldots, b_n^k$ and w_i^k is the weight of behavior b_i^k. Each agent (k-th) can explicitly ssigns a value to the maximal total of the weights of all its behaviors. The W_i^k is the maximum of the weight of behavior b_i^k. The maximum total of the weights of its behaviors in the k-th agent must be less than W^k. ($W^k \geq \sum_{i=1}^{n} w_i^k$), where $w_j^k - 1$ is 0 if w_j^k is 0. The W^k may depend on agents. In fact, W^k corresponds to the upper limit of the ability of each agent and may depend on the performance of the underlying system, including the processor.

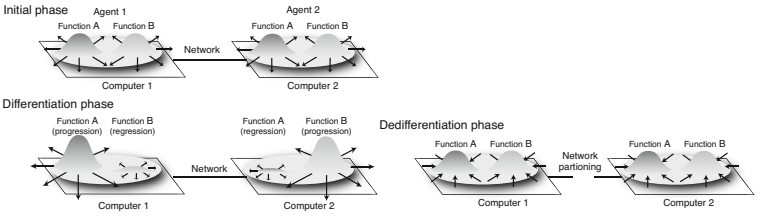


Fig. 1 Differentiation mechanism for software configuration

3.2 Function Invocation

When an agent wants to execute a behavior, it needs to select one of the available behaviors ($b_i^j, \ldots, b_i^m$), even if it has the behavior, according to the values of their weights. This involves three steps.

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

iii)The k-th agent delegates the selected agent to execute the behavior b_i^l and waits for the result from the l-th agent.

There is no universal selection function, ϕ, for mapping from the weights of behaviors to at most one appropriate behavior like that in a variety of creatures. Instead, the approach permits agents to use their own evaluation functions, because the selection of behaviors often depends on their applications. For example, one of the simplest evaluation functions makes the agent that wants to execute a behavior select one whose weight has the highest value and whose signature matches the wanted behavior if its database recognizes one or more agents that provide the same behavior, including itself.

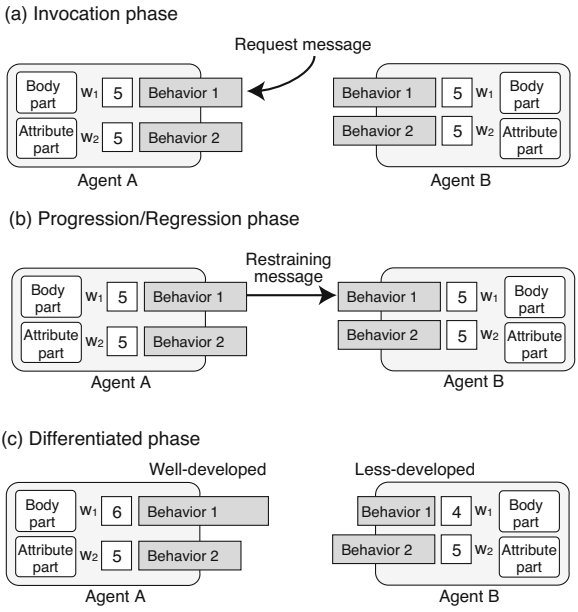


Fig. 2 Differentiation mechanism for agent

3.3 Differentiation

Our differentiation mechanism consists of two phases. The first involves the progression of behaviors in three steps.

i) When an agent (k-th agent) receives a request message from another agent, it selects the behavior (b_i^k) specified in the message from its behavior part and dispatches the message to the selected behavior. It executes the b_i^k behavior and returns the result.
ii) The k-th agent increases weight w_i^k of the b_i^k behavior.

iii)The k-th agent multicasts a restraining message with the signature of the behavior, its identifier (k), and the behavior's weight (w_i^k) to other agents.

When behaviors are internally invoked by their agents, their weights are not increased. If the total weights of the agent's behaviors, $\sum w_i^k$, is equal to their maximal total weight W^k, it decreases one of the minimal (and positive) weights (w_j^k is replaced by $w_j^k - 1$ where $w_j^k = \mathbf{min}(w_1^k, \ldots, w_n^k)$ and $w_j^k \geq 0$). Although restraining messages correspond to the diffusion of cAMP in differentiation, they can explicitly carry the weights of the agents that send them to reduce the number of restraining messages, because they can be substituted for more than one retaining message without weights. The second phase supports the retrogression of behaviors in three steps.

i) When an agent (k-th agent) receives a restraining message with regard to b_i^j from another agent (j-th), it looks for the behaviors ($b_m^k, \ldots b_l^k$) that can have the signature specified in the received message.
ii) If it has such behaviors, it decreases their weights ($w_m^k, \ldots w_l^k$) in its database and updates the weight (w_i^j) in its database.
iii)If the weights ($w_m^k, \ldots, w_l^k$) are under a specified value, e.g., 0, the behaviors ($b_m^k, \ldots b_l^k$) are inactivated.

4 Evaluation

Although the current implementation was not constructed for performance, we evaluated that of several basic operations in a distributed system where eight computers (Intel Core 2 Duo 1.83 GHz with MacOS X 10.6 and J2SE version 6) were connected through a giga-ethernet. The cost of transmitting a heartbeat or restraining message through UDP multicasting was 11 ms. The cost of transmitting a request message between two computers was 22 ms through TCP. These costs were estimated from the measurements of round-trip times between computers. We assumed in the following experiments that each agent issued heartbeat messages to other agents every 100 ms through UDP multicasting.

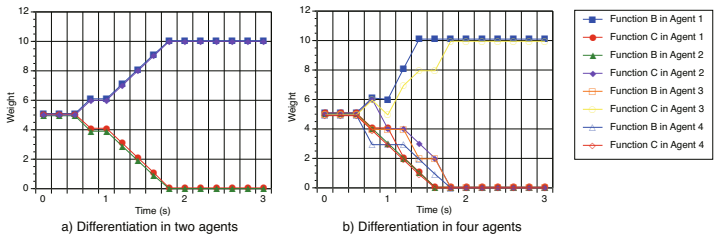


Fig. 3 Degree of progress in differentiation-based adaptation

Each agent had three behaviors, called A, B, and C. The A behavior periodically issued messages to invoke its B and C behaviors or those of other agents every 200 ms and the B and C behaviors were null behaviors. Each agent that wanted to execute a behavior, i.e., B or C, selected a behavior whose weight had the highest value if its database recognized one or more agents that provided the same or compatible behavior, including itself. When it invokes behavior B or C and the weights of its and others behaviors were the same, it randomly selected one of the behaviors. We assumed in this experiment that the weights of the B and C behaviors of each agent would initially be five and the maximum of the weight of each behavior and the total maximum W^k of weights would be ten. Figure 3 presents the results we obtained from the experiment. Both diagrams have a timeline in minutes on the x-axis and the weights of behavior B in each agent on the y-axis. Differentiation started after 200 ms, because each agent knows the presence of other agents by receiving heartbeat messages from them.

5 Conclusion

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

References

1. Agha, G.A.: ACTORS: A Model of Concurrent Computation in Distributed Systems. The MIT Press (1986)
2. Bigus, J.P., Schlosnagle, D.A., Pilgrim, J.R., Mills, W.N., Diao, Y.: ABLE: A toolkit for building multiagent autonomic systems. IBM Systems Journal 41(3), 350–371 (2002)
3. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press (1999)
4. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press (2004)
5. Herrman, K.: Self-organizing Replica Placement - A Case Study on Emergence. In: Proceedings of 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2007), pp. 13–22. IEEE Computer Society (2007)
6. Snyder, P.L., Greenstadt, R., Valetto, G.: Myconet: A Fungi-Inspired Model for Superpeer-Based Peer-to-Peer Overlay Topologies. In: Proceedings of 3rd IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2009), pp. 40–50 (2009)

Towards Characterizing Distributed Complex Situation Assessment as Workflows in Loosely Coupled Systems

Costin Bădică, Claudine Conrado, Franck Mignet, Patrick de Oude, and Gregor Pavlin

Abstract. This paper introduces challenges in contemporary situation assessment using collaborative inference and discusses solutions that are based on workflows between distributed processing nodes. The paper exposes the necessary conditions that workflows have to satisfy in order to support accurate situation assessment and provides a systematic approach to verification of the workflows. In particular, we emphasize the link between the complexity of the domain and the complexity of the workflows in terms of data and control coupling. With the help of graphical representations, we characterize the complexity of the domains and identify critical relations that have to be captured by collaborating processes in a workflow supporting correct situation assessment.

1 Introduction

Situation assessment is a critical capability in contemporary applications such as crisis management, security operations and control of complex systems. Often situation assessment requires inference about phenomena that cannot be observed directly (hidden phenomena) but that are critical for decision making. In such cases, the inference process is based on (i) large quantities of heterogeneous observations and (ii) domain models that capture the relations between observations and hidden phenomena. In general, situation assessment in these applications is inherently

Costin Bădică
Software Engineering Department, Faculty of Automatics, Computers and Electronics, University of Craiova, Bvd.Decebal, Nr.107, Craiova, RO-200440, Romania
e-mail: cbadica@software.ucv.ro

Claudine Conrado · Franck Mignet · Patrick de Oude · Gregor Pavlin
D-CIS Lab / Thales Research & Technology Netherlands, P.O. Box 90, NL 2600 AB DELFT, The Netherlands
e-mail: {Claudine.Conrado,Franck.Mignet,Patrick.deOude, Gregor.Pavlin}@d-cis.nl

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 115–127.
springerlink.com

complex and requires substantial domain knowledge and processing power. A single expert cannot solve such problems, while construction of reliable, fully automated solutions is usually intractable (see a more thorough discussion on this topic in [14]). Therefore, it seems reasonable to solve complex situation assessment problems through collaboration of multiple experts and/or automated processes, each supporting analysis services based on specific domain knowledge. In other words, the solutions are based on workflows between experts and automated processes; the experts and processes can be viewed as processing modules which exchange analysis products/estimates and observations. The performance of such collaborative solutions critically depends on the workflows in which the messages that carry analysis results are transported. These workflows have to support reasoning that results in correct conclusions about the domain and must be rigorous from the control flow perspective [16].

In this paper we show that the correctness of the situation assessment processes critically depends on the workflow structure, which must reflect the domain complexity, i.e. the dependencies between the observable and hidden phenomena of interest. Moreover, tractable solutions require loosely coupled processes. We introduce directed graphs representing qualitative aspects of causal processes, which facilitate a systematic characterization of problems. In this way, we can investigate whether loosely coupled solutions to a specific situation assessment problem exist and determine the critical dependencies that have to be captured by the underlying workflow in which assessment takes place.

Beside the inference challenges, the workflows have to be rigorously analyzed according to desired properties including termination, deadlock-freedom, safety and soundness. By employing standardized approaches to workflow modeling [7] and analysis using Petri net representation (see for example WF-nets), we map situation assessment processes to rigorous workflow models. In this way, systematic analysis using the available tools is enabled.

We use a running example from the crisis management domain throughout this paper to illustrate the concepts and challenges introduced above.

2 Situation Assessment and Causal Processes

We limit our discussion to the situation assessment problems whose inference process is based on (i) knowledge about observable phenomena and (ii) generic domain knowledge in form of models (mental or AI based) capturing the relations between observed and hidden phenomena. An assessment process results in correct conclusions about the hidden phenomena (i.e., conclusions that correspond to the *actual situation*) if the domain models correctly describe the relations between the relevant phenomena and the inference correctly considers the relations in the models.

Moreover, hidden and observable phenomena of interest are often outcomes of causal processes. In the present work, we take the view of a domain as a *combination of dependent causal processes*. A similar view has been adopted by researchers in the field of qualitative reasoning, e.g. within the framework of Qualitative Physics

[4], to describe phenomena in the physical world. A physical system is modeled as a set of causal processes at work in the domain of interest and their interactions (yielding the physical system's *structural* representation). Moreover, phenomena of interest in the physical system are realized through the composition of causal processes, each with given input and output variables. Combinations of causal processes are also relevant in the Functional Representation framework discussed in [2]. The framework describes the causal processes that realize device functions and, more generally, it aims at providing a broader framework for human reasoning about the physical world and the role played by causal processes.

In order to illustrate the concepts and challenges discussed in this paper, we use an example from the crisis management domain, shown in Fig. 1. An incident at a chemical factory results in a gas leak which, due to the weather conditions, gives rise to an invisible gas plume spreading over a populated area. In order to assess the situation, the potential impact on the population and the associated costs need to be estimated. This requires substantial expertise and information about observable events. We have to estimate the extent of the plume by using knowledge about the leak, weather and the physical processes that result in the plume. The plume estimate in combination with knowledge about the population distribution and the physiological impact on the human body are used for the estimation of the possible risks and costs. In this example, we can identify a chain of dependent processes, namely outflow of the chemical at the leak, convection and advection leading to the gas plume, chemical reaction in the human body and associated costs. We call such a set of causal processes and their relations the "Domain Structure", which here corresponds to the *actual situation* in the domain (i.e. the ground truth).

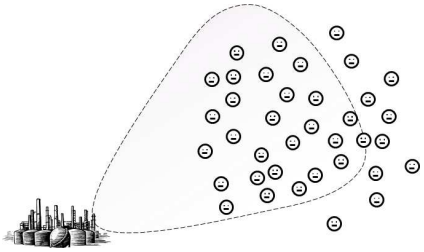


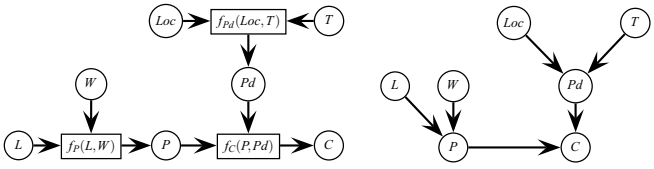
Fig. 1 An invisible plume in a chemical incident has to be estimated and its impact on the population has to be predicted

2.1 Graphical Domain Representations and Generic Process Models

The structural representation of the Domain Structure is based on graphs. We provide two types of representations, the *process* graphs and *causal* graphs.

The process graphs are directed bipartite graphs in which circles represent the phenomena and rectangles represent the processes. Directed links between the circles and the rectangles show the dependencies between the processes and phenomena. These graphs are similar to Petri-nets representing the processes in the domain. Fig. 2a shows a process graph describing the chemical incident in our example. This incident/domain can be decomposed into three distinctive processes, each involving the estimations of plume ($f_P(L,W)$), population distribution ($f_{Pd}(Loc,T)$) and cost ($f_C(P,Pd)$). The root nodes represent context phenomena, i.e., the boundary conditions of the overall process of interest.

The causal graphs, on the other hand, show the direct influences between the phenomena that materialize in a Domain Structure, but do not explicitly represent the processes. The processes are implicitly encoded in the relations between the variables. This type of graphs is analogous to the qualitative models of physical processes in [9] and the qualitative representation of causal probabilistic models (i.e. Bayesian networks) [12, 13]. Fig. 2b shows a causal graph describing our running example.



(a) process graph (b) causal graph

Fig. 2 The process graph and the causal graph describing the Domain Structure for the chemical incident example

As suggested by the process graph in Fig. 2a, the processes in a given domain can be viewed as *functions*, which allow us to make conclusions about the hidden phenomena. For example, by knowing the process of plume formation, i.e. the function $f_P(L,W)$, we can estimate the extent of the plume P. Usually such knowledge is not perfect. Instead, we use models of processes that capture the main tendencies, i.e., they are abstractions of the true functions.

Furthermore, the representation of the phenomena of interest can be very diverse, depending on the used process models. In case of probabilistic models, random variables are used and probability distributions over a fixed set of phenomena are communicated between the models. In traditional qualitative reasoning approaches, logical models are used to manipulate deterministic process representations [9]. In case humans are estimating the situation using mental models, the inputs and outputs of the inference process can be e.g. graphical, textual or audio. In our example, the cost estimation could be carried out by experts using simple models. Based on leak

and weather information, an expert uses a simple model to draw a plume on top of a map, which is then laid over a map of the current population distribution (obtained by using information on location, time and a model for prediction of population distribution at different times). The expert simply estimates the number of exposed people by counting the people in the area within the plume contour. The final step is to estimate the impact on an average person and the associated cost, and then multiply this with the estimated number of exposed people to estimate the total cost.

It is important to emphasize that the above graphical representations do not provide the assessment mechanisms, but they merely serve the analysis of the problems. However, *in order to support accurate situation assessment, an important condition is that the process graph must be captured in the workflows between processing nodes*. Moreover, we assume arbitrary process models, either mathematical or mental. The main requirements are that (i) there exists a notion of relations between clearly defined phenomena (i.e. variables) and (ii) the models capture those relations with sufficient accuracy and resolution.

3 Locality of Causal Relations and Domain Complexity

The graphical representation of the processes in the domain facilitates the complexity analysis of the estimation problems. One of the key concepts required to understand the complexity of the domain is *locality of causal relations*, which can be investigated by inspecting the functions associated with causal relations. A function associated with a causal relation has the generic form $Y = f_Y(\mathbf{Pa}(Y))$, where **Pa** are the parents (causes) of variable Y (effect) and the set of variables $\{Y, \mathbf{Pa}(Y)\}$ are considered *local* to the causal process that the function represents. Because the function uses all the direct causes of the effect as its arguments, we can define independence of events, as in [15]:

Definition 1 (Independence of Events). When the direct causes of an effect occurred then the occurrence of the indirect causes no longer has anything to do with the effect. In other words, direct causes *screen off* the indirect causes from the effect.

According to Definition 1, variables in the domain can be localized based on the direct causes of effects. To illustrate this, consider the causal graph in Fig. 2b. In this example, the causal influence of weather W and leak L on the costs C is mediated by the plume P, i.e. the plume is a direct cause of the cost C whereas weather W and leak L are indirect causes. From the reasoning perspective and according to Definition 1, if the state of the plume P is known, then no information on the weather or leak improves the information about the cost C. In other words, C is rendered independent of W and L given the state of P. Moreover, the local variables $\{W, L, P\}$ and the local variables $\{P, Pd, C\}$ can easily be separated from each other by only sharing the overlapping, mediating variable P.

Note, however, that weather W and leak L become *dependent* (induced dependence) when the state of P is known. This dependence can be shown with the process graph depicted in Fig. 2a. The rectangle that is connected to the variables W, L

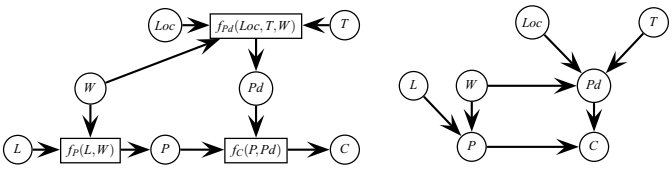
and P corresponds to the function $P = f_P(L,W)$. Knowing the state of P, information on the argument L will tell us something about the state of the argument W in the function $f_P(L,W)$.

From Fig. 2, it can also be seen that the in-degree[1] of a node (variable) determines the cardinality of the arguments of a function. The greater the in-degree of a process node, the greater the number of arguments that have to be considered in the function describing the process. Functions with more arguments are likely to be more complex and the collection of the required information is more elaborate.

Additionally, a higher in-degree and out-degree of nodes in a causal or process graph will more likely result in loops[2]. Loops imply more complex models and synchronization of events, which has consequences for the assessment processes. For example, if our example took place in a city where the population distribution depends on the weather (e.g., a city at the seaside where more people go to the beach in case of good weather), we would need a more complex model, as depicted in Fig. 3. In this example, the weather W influences the population distribution Pd, as well as the plume P, resulting in a loop. In order to deal with this loop, the instantiation of the weather W must be synchronized such that the function $C = f_C(P,Pd)$ is based on the same value of W.

Note that in the case of Bayesian networks, loops in the model are eliminated by constructing a secondary probabilistic structure where cliques form hyper variables [5, 10]. However, in a general domain where the underlying models are not necessarily probabilistic this is not a straightforward procedure, due to the non-uniformity of the processes involved.

As we will show in the following sections, the connectivity of the graphs has direct consequences for the construction of workflows that support correct collaborative assessment.



(a) process graph (b) causal graph

Fig. 3 Process and causal graphs describing the Domain Structure for the more complex chemical incident example

[1] *In-degree* is the number of directed links pointing to a certain node whereas *out-degree* is the number of directed links pointing away from a certain node.

[2] *Loops* occur if there exist at least two unique paths between any two nodes in the graph. *Cycles*, on the other hand, exist if by following a certain path we return to a previously visited node. Note that the causal graphs are *acyclic* graphs.

4 Distributed Assessment

In contemporary assessment problems, e.g. large-scale crisis management, the acquisition of the relevant information from various sources as well as its interpretation require very rich domain knowledge and often huge processing capabilities. Centralized solutions are often not tractable due to communication and processing bottlenecks that arise when huge amounts of data have to be transported to and processed at a single analysis center. In addition, it is likely that the maintenance of the processing and communication mechanisms becomes complex. Finally, centralized data collection and processing are often not desired due to privacy/sensitivity issues.

In such settings, the distribution of acquisition and assessment efforts throughout multiple collaborating processes seems very promising and often the only viable way to proceed. Since the domains can be described as a composition of interdependent processes, it seems reasonable to distribute the assessment task throughout a *system of communicating modules*, each using a model of a given process in the domain. The assessment of a critical state in such a system is then based on the composition of complementary modules.

Given the distributed approach above, it becomes crucial to *investigate the circumstances under which we can efficiently achieve correct assessment, i.e. correct interpretation of the information stemming from different components at different locations and times*.

Key to efficient distributed assessment is loose coupling of the distributed processes. Interactions in collaborative assessment processes involve sharing of data (data coupling) and partial synchronization of inference processes in different modules (control coupling).

4.1 Composing Assessment Processes

Each module in the system provides a specific transformation between different types of information and it must receive all the inputs required to produce the output. The overall assessment is reduced to the communication of specific estimates and observations between pairs of modules, each processing information about the subset of the relevant phenomena in the domain. This is a data-driven approach, with providers and consumers of information communicating in a peer-to-peer fashion; as soon as a module updates its estimate about a variable's states, this is communicated to other modules whose inference also depends on that variable. Upon receipt of such a message, a local inference process using a specific model is triggered and other hidden states/phenomena are estimated. Fig. 4 shows a modular system implementing distributed assessment in our example. Module M_1 receives from external sources information about the leak L and the weather W, which is used to estimate the plume P. This estimate is then passed on to module M_2, which also receives an estimate of the population distribution Pd from M_3. M_3 uses the knowledge on the location Loc and the current time in combination with a population model to estimate Pd. M_2 outputs a cost estimate C of the current situation.

If each module supports estimation of any variable in its local function (i.e., if the function can be inverted), then the same constellation of modules can be used for different queries, e.g. questions about the values of different variables. For example, the three modules in Fig. 4 could infer the location of the leak L by knowing the extent of the cost C. In such a case, M_2 would use the output from M_3 as input and the information about the cost to estimate the plume P, which would be passed to M_1. M_1 would invert function f_P describing the plume evolution to estimate the leak L, by using the weather W as input.

The example system in Fig. 4 is loosely coupled. Namely, each pair of modules exchanges information on a single phenomenon (i.e. the domain models of each pair share a single variable). In addition, the control coupling is also simple and does not require any centralized processing. In this data-driven approach, a local assessment process is triggered *locally*, as soon as all the critical inputs are available.

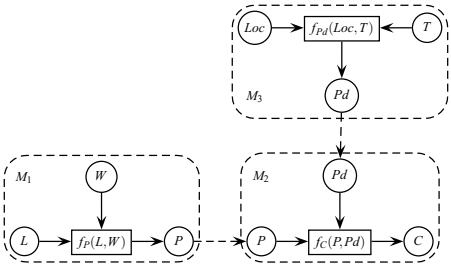


Fig. 4 A modular approach to situation assessment

4.2 Domain Structures and Dependencies in Distributed Systems

In order to achieve correct assessment in distributed settings, the collaborative system has to support globally consistent inference; i.e. any conclusions made by the distributed system must correctly consider all the relevant observations/information about the domain that have been collected/produced by different processes, which results in correct inference. *The necessary condition for this is that the system of collaborating modules correctly considers the dependencies between the phenomena in the Domain Structure.* This means that a reasoning module M_i must take into account *all* phenomena that *directly* influence the phenomenon it is estimating.

For example, module M_2 in Fig. 4 correctly reflects the direct dependencies between P, Pd and C that exist in the domain, as captured by the Domain Structure in Fig. 2. Moreover, the system in Fig. 4 supports correct reasoning if the whole domain is captured by the Domain Structure in Fig. 2.

For the estimation of the domain shown in Fig. 3, however, the system in Fig. 4 is inappropriate, as it does not have the capacity to reason about the influence of the weather on the population distribution. This mismatch could result in severe impact on the assessment accuracy. For example, we assume that the plume estimator M_1 uses the weather information and obtains an accurate prediction of the plume,

which spreads over the beach area. Moreover, given the wind, time and the high temperature, a significant portion of the citizens is at the seaside. The population estimator M_3, differently from the plume estimator M_1, uses an average population model which does not explicitly consider the weather, concluding that most people are indoors. Therefore, the cost estimator M_2 processes the estimates of plume and population distribution, estimating the cost as *small*. Obviously, this conclusion is wrong and could have severe consequences for the decision making process.

The system shown in Fig. 5, on the other hand, is appropriate for the estimation of the domain shown in Fig. 3, given that the local models correctly relate the variables. In this case, module M_3 would take the weather W into account, so it would conclude that a high population distribution at the beach is very likely.

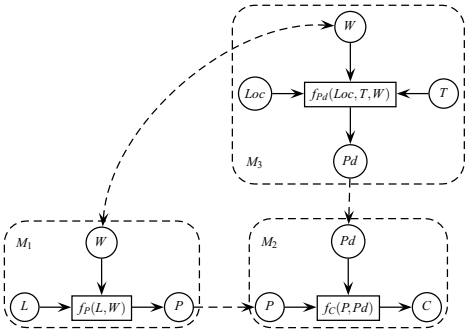


Fig. 5 A modular approach to situation assessment that correctly considers the dependency of population distribution on the weather

In Fig. 5, we now have to make sure that both module M_1 and module M_3 use the same information on the weather W. This is achieved with duplication, where the source of information on W should make sure that both modules are updated simultaneously. Module M_2 uses P from module M_1 and Pd from module M_3, and it needs both inputs to be present to perform its local assessment. This can be seen as a form of control coupling, where the inputs that are presented lastly control the start of the realization of the task, i.e. function f_C. In addition, control coupling is introduced through a synchronization mechanism that ensures M_2 uses estimates P and Pd based on the same value of W.

The system in Fig. 5 is thus more complex in terms of data and control coupling than the system in Fig. 4. In general, the data and control coupling depend on the Domain Structure complexity: the more complex the Domain Structure, the more complex the data and control couplings. *Distributed systems can efficiently be implemented, especially if the Domain Structure has a tree-topology with low in-degree.*

While this paper does not discuss all data and control coupling aspects, it is easy to see that coupling can have a significant impact on the performance in terms of throughput but also on the design complexity, depending on the choice of

synchronization mechanism. Additional insight on the problem of coupling, relevant for the presented assessment challenges, can be obtained from the field of distributed computer simulations, in particular discrete event simulation [1, 3, 6, 11, 17].

In the next section, we discuss workflows which are an essential tool to organize correct sharing of information between different modules and deal with the complexity of data and control coupling.

5 Rigorous Workflows: A Basis for Sound Distributed Assessment

Globally consistent assessment is achieved by creating workflows between compatible processing services, each implementing a specific function that supports transformation between different types of information, referred to as *composition of processes*. In order to achieve globally consistent distributed processing, it is crucial to have (i) a rigorous definition of workflows and (ii) the knowledge of the main dependencies in the domain that the processes are estimating.

A workflow is represented as a structured set of coordinated tasks and information exchanges defined to carry out a well-defined business process in organizations [16]. Collaborative problem solving in heterogeneous, unpredictable and dynamic environments can be achieved by providing an increased flexibility in workflow formation that goes far beyond the classical approach of static and centralized workflow definitions. Examples based on empirical analysis of real experiences can be given in the area of organizational adaptation to disasters [8].

Workflows for distributed complex situation assessment are dynamically created using dynamic discovery of links between weakly coupled processes [14]. Dynamic workflow formation exploits the dependencies present in the structure of the problem domain. This means that such workflows are not a priori defined; rather, they are formed dynamically depending on the Domain Structure, on the availability of appropriate resources, as well as on the requirements posed by a specific complex situation assessment problem.

More formally, the Domain Structure guides the dynamic formation of a workflow model. In particular, from the causal graph characterizing the Domain Structure, we can derive a *standard workflow model* (SWM). According to the terminology introduced in [7], a SWM is composed of *activities* – represented as rectangles – and *control-flow constructs* – including AND-joins, AND-splits, OR-joins and XOR-splits. Additionally, basic control-flow constructs of a SWM contain input and final activities, as well as sequences of activities.

In what follows we consider our running example, in particular focusing on the process graph shown in Fig. 3 that contains a loop. We can define a mapping from this process model to a SWM by considering two types of activities.

Firstly, for each phenomenon modeled as a variable Y that is influenced by phenomena $X_1, \ldots, X_n$ according to a local function $f_Y(X_1, \ldots, X_n)$, we define an workflow activity for the realization of the local reasoning process corresponding to the evaluation of function $f_Y(\ldots)$. This activity is enabled when the values of variables

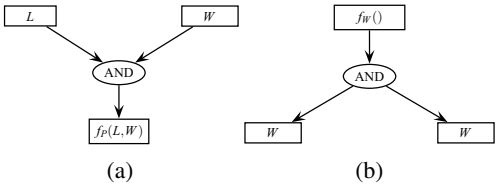


Fig. 6 Local mappings of process graphs to parts of SWM

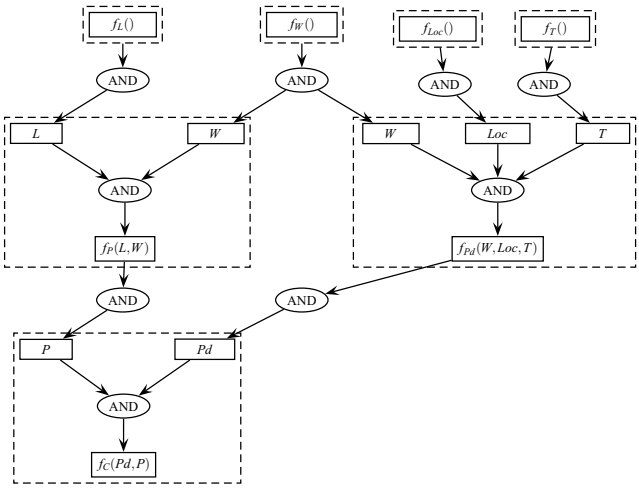


Fig. 7 Sample SWM for a situation assessment problem

$X_1,\ldots,X_n$ are available. Such a situation is shown in Fig. 3 for variable P depending on variables L and W according to function $f_P(L,W)$, while the corresponding part of the SWM is shown in Fig. 6a.

Secondly, for each phenomenon modeled as variable X that influences variables $Y_1,\ldots,Y_m$, we must include in the SWM the activities of passing the value computed by the local process that evaluates X according to the function $f_X(\ldots)$ to each of the local processes that will need X for evaluating variables $Y_1,\ldots,Y_m$. Such a situation is shown in Fig. 3 for variable W, which must be propagated to the modules that evaluate variables P and Pd. The corresponding part of the SWM is shown in Fig. 6b.

Now, by the systematic application of these two mapping rules, the process graph shown in Fig. 3 can be mapped to the SWM shown in Fig. 7. Note that root nodes representing context phenomena (i.e., L, W, Loc and T) correspond to the input activities of the SWM, i.e. SWM activities labeled with function names without variables, e.g. $f_L()$, $f_W()$ and so on. Moreover, the resulting SWM for this example contains only AND-joins and AND-splits, so it is considerably simpler than the most general SWM introduced in [7]. More complex situation assessment problems are likely to generate more complicated SWMs.

6 Conclusions

The main objectives of this paper are (i) to provide a framework that allows analysis of complex situation assessment problems, (ii) to expose the necessary conditions that workflows have to satisfy in order to support accurate situation assessment in collaborative systems and (iii) to provide a workflow formalization that allows systematic verification of complex workflows. In particular, we emphasize the link between the complexity of the domain and the complexity of the workflows in terms of data and control coupling.

Firstly, we show that the correctness of the situation assessment processes critically depends on the workflow structure, which must reflect the dependencies between the observable and hidden phenomena of interest. On the other hand, tractable solutions require loosely coupled processes. We introduce directed graphs representing qualitative aspects of causal processes, which facilitate investigation of the problem complexity. With such graphs, we can systematically analyze whether loosely coupled solutions to a specific situation assessment problem exist and determine the critical dependencies that have to be captured by the underlying workflow in which assessment takes place.

Secondly, the resulting workflows are described in a way that supports rigorous analysis of their properties and verification of their correctness (terminating conditions, deadlocks, and so on).

In our discussions we assumed arbitrary modeling and inference techniques. Note that such techniques can have consequences for the construction of workflows. For example, in the special case where the modules represent cliques in a causal Bayesian network and Junction tree algorithm is used for inference, the complexity of control coupling is reduced but the complexity of data coupling is increased. The synchronization in this case is reduced to a local rule that specifies which variables can be marginalized out. However, for more general inference techniques lacking a rigorous theoretical basis or involving non invertible functions/processes, the complexity of the control coupling in workflows cannot be reduced. Instead, the loops in the Domain Structure have to be solved by introducing synchronization of information flows through converging paths throughout the workflow. In those cases, techniques similar to the cut-set approach used for loopy Bayesian networks can be applied.

This paper presents the initial results of ongoing research and the discussion focuses on predictive reasoning only. Future work will focus on the relations between the workflows and different types of reasoning, e.g. diagnostic and predictive inference. In addition, following [7], a more in-depth investigation regarding the mapping of the resulting SWM to a Petri net – and its rigorous analysis for desired properties using workflow analysis tools [16] – will be carried out.

Acknowledgements. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement n261817 (BRIDGE project (www.bridgeproject.eu)) within the Security Programme SEC-2010.4.2-1: Interoperability of data, systems, tools and equipment.

References

1. Bhatt, E., Fujimoto, R., Ogielski, A., Perumalla, K.: Parallel simulation techniques for large scale networks. IEEE Communications Magazine 36, 42–47 (2002)
2. Chandrasekaran, B.: Functional representation and causal processes. Advances in Computers 38, 73–143 (1994)
3. Deelman, E., Szymanski, B.K., Caraco, T.: Simulating lyme disease using parallel discrete event simulation. In: Proceedings of the Winter Simulation Conference, pp. 1191–1198 (1996)
4. Forbus, K.D.: Qualitative physics: Past, present, and future. In: Exploring Artificial Intelligence, ch. 7, pp. 239–296. Morgan-Kaufmann Publishers, Inc., San Francisco (1988)
5. Jensen, F.V., Nielsen, T.D.: Bayesian Networks and Decision Graphs, 2nd edn. Springer (2007)
6. Karimabadi, H., Driscoll, J., Dave, J., Omelchenko, Y., Perumalla, K., Fujimoto, R., Omidi, N.: Parallel Discrete Event Simulations of Grid-Based Models: Asynchronous Electromagnetic Hybrid Code. In: Dongarra, J., Madsen, K., Waśniewski, J. (eds.) PARA 2004. LNCS, vol. 3732, pp. 573–582. Springer, Heidelberg (2006)
7. Kiepuszewski, B., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of control flow in workflows. Acta Informatica 39(3), 143–209 (2003)
8. Kreps, G.A., Bosworth, S.L.: Organizational adaptation to disaster. In: Handbook of Disaster Research, ch. 17, pp. 297–315. Springer (2007)
9. Kuipers, B.: Qualitative simulation. Artificial Intelligence 29, 289–338 (2001)
10. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems, pp. 415–448 (1990)
11. Lukovszki, C., Szabó, R., Henk, T.: Performance evaluation of a hybrid atm switch architecture by parallel discrete event simulation. Informatica (Slovenia) 24(2) (2000)
12. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
13. Pearl, J.: Causality: Models Reasoning and Inference. Cambridge University Press (2000)
14. Penders, A., Pavlin, G., Kamermans, M.: A collaborative approach to construction of large scale distributed reasoning systems. International Journal on Artificial Intelligence Tools 20(6), 1083–1106 (2011)
15. Spirtes, P., Glymour, C., Scheines, R.: Causation, Prediction, and Search, 2nd edn. MIT Press (2000)
16. van der Aalst, W.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
17. Wieland, F.: Parallel simulation for aviation applications. In: Winter Simulation Conference, pp. 1191–1198 (1998)

A Trust-Based Approach for a Competitive Cloud/Grid Computing Scenario

Fabrizio Messina, Giuseppe Pappalardo, Domenico Rosaci, Corrado Santoro, and Giuseppe M.L. Sarné

Abstract. Cloud/Grid systems are composed of nodes that individually manage local resources, and when a client request is submitted to the system, it is necessary to find the most suitable nodes to satisfy that request. In a competitive scenario, each node is in competition with each other to obtain the assignment of available tasks. In such a situation, it is possible that a node, in order to obtain the assignment of a task, can lie when declaring its own capability. Therefore, lying nodes will need to require the collaboration of other nodes to complete the task and consequently the problem arises of finding the most promising collaborators. In such a context, to make effective this selection, each node should have a trust model for accurately choosing its interlocutors. In this paper, a trust-based approach is proposed to make a node capable of finding the most reliable interlocutors. This approach, in order to avoid the exploration of the whole node space, exploits a P2P resource finding approach for clouds/grids, capable of determining the admissible region of nodes to be considered for the search of the interlocutors.

1 Introduction

Nowadays, the offer of on-demand computing resources is rapidly increasing, changing the way in which users deal with special purpose computing requirements. Consequently, approaches such cloud computing [2] and grid computing [1] are becoming the most widespread and leading concepts in the field of networked

Fabrizio Messina · Giuseppe Pappalardo · Corrado Santoro
Dipartimento di Matematica e Informatica, Università di Catania, V.le Andrea Doria 6, 95125 Catania, Italy
e-mail: {fmessina,pappalardo,santorog}@dmi.unict.it

Domenico Rosaci · Giuseppe M.L. Sarné
DIMET, University "Mediterranea" of Reggio Calabria, Via Graziella, Feo di Vito 89122 Reggio Calabria (RC), Italy
e-mail: {domenico.rosaci,sarne}@unirc.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 129–138.
springerlink.com

distributed systems. These systems are composed of nodes that individually manage local resources, and when a client request is submitted to the system, it is necessary to find the most suitable nodes to satisfy that request. In a collaborative scenario the main problem is that the user request must be fulfilled as soon as possible, determining the nodes with the necessary resources, while in the case of a competitive cloud/grid system another issue must be considered. In a competitive scenario, where clients pay a price to obtain a service, each node is in competition with each other to obtain the assignment of available tasks. In such a situation, it is possible that a node can lie when declaring its own capability. A lying node will need to require the collaboration of other nodes to complete the task, paying a price, and thus the problem arises of finding the most promising collaborators. Since the nodes are in competitions, they might be fraudulent or malicious when collaborating with other nodes. Consequently, each node should have a trust model for accurately choosing its interlocutors. In the context of the e-services, trust is defined as:"the quantified belief by a truster with respect to the competence, honesty, security and dependability of a trustee within a specified context" [4]. When two agents interact with each other, one of them (the truster) assumes the role of a service requester and the other (the trustee) acts as an e-service provider. In this context, while *reliability* is a subjective measure, *reputation* is a measure of the trust that the whole community perceives with respect to a given trustee. Several reliability and reputation models have been proposed in the past for representing both reliability and reputation [4, 9, 10]. In particular, we have proposed the RRAF model [3, 8] to suitably combining these two measures into a unique, global trust measure. In RRAF, each node selects among all the nodes the most promising collaborators based on such a trust measure. However, when the size of the system becomes large, as in the case of many cloud/grid systems, this selection task becomes impracticable.

Given these premises, in this paper we propose to modify the RRAF approach to make it applicable to large cloud/grid systems. Our idea is that of introducing the possibility, for a node that has to choose its interlocutors, of limiting the search space to only those nodes that declare to have the necessary resources for realizing the collaboration. To this aim, we propose to use a technique, called SW-HYGRA (standing for Small World-HYperspace Grid Resource Allocation) [7], which organizes the servers/computing nodes, representing peers, in an overlay network featuring certain characteristics aiming at suitably making the resource finding process effective and efficient. The basic model of SW-HYGRA is to employ an overlay construction algorithm which exploits the resource status similarity, i.e. peers featuring a similar amount of resource availability tend to be interconnected - by means of the links of the overlay - thus forming clusters which, in turn, are connected together by means of few long links. Such an organization resembles the classical model of small-world networks [12]. Some experiments we have realized show that the use of SW-HYGRA in combination with a trust-based selection of the interlocutors introduce in a competitive environment a significant advantage for a node, with respect to other nodes that select their interlocutors based on the sole resource declaration.

The remaining of the paper is organized as follows. In Section 2 we introduce the competitive cloud/grid scenario we deal with. Section 3 describes our reliability and reputation model, while 4 introduces the technique for finding suitable nodes with SW-Hygra. An experimental evaluation of the proposed approach is provided in Section 5. Finally, in Section 6 we draw our conclusions and discuss possible developments of our ongoing research.

2 The Competitive Cloud/Grid Scenario

Our approach deals with a cloud/grid system, composed by nodes that provide a set of services to clients, and that are in competitions to obtain the tasks requested by the clients. In order to model such a competition, we suppose that when a client needs a service, he sends a request to a *Task Allocator* TA, which assigns the request to a node. For sake of simplicity, the assignment of the client's request to a node is performed by TA at pre-determined temporal steps. When a new step begins, TA examines all the service requests submitted by clients, and assigns each request to the node considered the most suitable based on the effectiveness shown by it in the past. When the assignment is done, the client must pay a given *service price sp* to the selected node to obtain the service. During each temporal step, the nodes of the system can interact with each others, in order to exchange information. The interactions among nodes follow this protocol:

- A node a, in order to provide a client with a service requiring a resource amount $\overline{q}$, may decide to search the collaboration of another node b, belonging to the admissible region $S(\overline{q})$ (see Section 4). Before requiring this collaboration, a could ask a third node c for a recommendation about the expertise of b. This recommendation is an evaluation of the *Quality of Service* (QoS) generally associated with the services provided by b. The node c can accept or refuse to give the recommendation. If it accepts, then a must pay a given *reputation price rp* to b. The obtained recommendations can be used by a for updating its internal *reputation model* (see Section 3). In words, a uses the gossips coming from the other nodes in order to understand what is the reputation of b in the community.
- At the end of the step, TA *i)* directly asks a feedback to the client about his evaluation of the quality of each service that a provided him during the step and *ii)* provides this client's feedback to a. The feedback informs a about the quality of the contributions given by the contacted nodes. This way, a can use the feedback to update its internal *trust model*.

3 The Reliability-Reputation Model

This section presents a trust model dealing with the previously introduced scenario. Moreover, we present a methodology for computing the trust measures involved in this scenario: reliability and reputation.

We denote by $\mathscr{A}$ the list containing all the nodes belonging to the cloud/grid system, and by a_i the i-th element of $\mathscr{A}$. A set of five mappings, denoted by SR_i, RR_i, R_i, β_i, and P_i is associated with each node a_i, where each mapping receives a node j as input and yields as output a different trust measure that the node a_i assigns to the node a_j. Each trust measure is represented by a real number belonging to the interval $[0,1]$, where 0 (1) is the minimum (maximum) value of trust. In particular:

$SR_i(j)$ represents the *service reliability* that the node a_i assigns to the services provided by the node a_j. We recall that the reliability represents the subjective measure of the trust that a node has in another node. The value 0 (1) means complete unreliability (reliability).

$RR_i(j)$ represents the *recommendation reliability* that a_i assigns to the *recommendations* provided by a_j. In other words, $RR_i(j)$ is a measure of how much the node a_i considers as reliable the suggestions coming from the nodes j about other nodes, i.e. it represents the reliability of the gossip coming from a_j.

$R_i(j,c)$ represents the *reputation* that the node a_i assigns to the node a_j, based on some recommendations coming from nodes of the community. Although the reputation is not based on a subjective evaluation of the node, it is not an objective measure, since each node a computes the reputation of another node b independently of how the other nodes compute it. Thus, we can say that the value $R_i(j)$ represents how the node a_i perceives the reputation of a_j in the community. The value 0 (1) means minimum (maximum) reputation.

$\beta_i(j,c)$, called *reliability preference*, represents the *preference* that a_i assigns to the usage of the reliability with respect to the reputation in evaluating a_j. In other words, when a_i computes the overall trust score to assign to a_j, it considers both the contributions of service reliability $SR_i(j)$ and reputation $R_i(j)$. The percentage of importance to give to the service reliability is represented by the value $\beta_i(j)$, while the percentage to assign to the reputation is 1-$\beta_i(j)$. In our framework, the mapping β_i is arbitrarily chosen by a_i following its personal strategy.

$P_i(j,c)$ represents the overall preference that ta_i assigns to a_j, based on both the reliability and reputation perceived by a_i.

Besides the five mappings described above, we define a mapping denoted by $RECC_i$, in order to represent the *recommendations* that the node a_i has obtained by the other nodes. Formally, $RECC_i$ is a mapping that receives two nodes a_j and a_k as input, and yields as output a recommendation $RECC_i(j,k)$ representing the recommendation that a_j provided to a_i about a_k (i.e., a real value ranging in [0,1]).

Then, the mappings are updated by a_i at each step, as follows:

- **Phase 1: Reception of the Recommendations.** a_i receives, at the current step, some recommendations by the other nodes, in response to previous recommendation requests. These recommendations are stored in the *RECC* mapping.
- **Phase 2: Computation of SR mapping.** The TA sends to a_i the feedbacks for each service s provided in the past step, where the contributions given by other nodes to a_i are evaluated. These feedbacks are contained in a mapping $FEED$,

where each feedback $FEED(s,j)$ is a real number belonging to $[0,1]$, representing the quality of the collaboration that the node a_j provided to the node a_i concerning the service s. A feedback equal to 0 (1) means minimum (maximum) quality of the service. Basing on these feedbacks, a_i updates its mappings SR and RR. Specifically, we choose to compute the current reliability shown by a_j in its collaboration with a_i by averaging all the feedbacks concerning a_j. Therefore, denoting by $Services(j)$ the set of services provided by a_i with the collaboration of a_j at the previous step, the current service reliability $sr(j)$ shown by a_j is computed as

$$sr(j) = \frac{\sum_{s \in Services(j)} FEED(s,j)}{Services(j)}$$

At each new step, this current reliability is taken into account for updating the element SR_i. We choose to compute this value by averaging the value of SR_i at the previous step and the current reliability computed at the new step. Thus:

$$SR_i(j) = \alpha \cdot SR_i(j) + (1-\alpha) \cdot sr(j)$$

where α is a real value belonging to $[0,1]$ and representing the relevance that a_i gives to the past evaluations of the reliability with respect to the current evaluation. In other words, α measures the importance given to the *memory* with respect to the current time. In an analogous way, the feedbacks are used to update the recommendation reliability RR. The current recommendation reliability of a_j at a given step is computed by averaging all the errors made by a_j in providing a recommendation. In words, if a_j recommended to a_i the node a_k with a recommendation $RECC(j,k)$, and the feedback for a_k concerning a service s is $FEED(s,k)$, the error made by a_j by its recommendation is $|RECC(j,k) - FEED(s,k)|$. By averaging all the errors concerning services of the category c, we obtain an evaluation of the current precision of a_j with respect to the recommendations relating to a_k, that is $(\sum_{s \in Services(k,c)} |RECC(j,k) - FEED(s,k)|)/Services(k)$. Finally, by averaging this precision on the set $Nodes(j)$ of all the nodes a_k evaluated by a_j in the previous step, we obtain the current recommendation reliability $rr(k)$:

$$rr(k) = \frac{1}{|Nodes(j)|} \sum_{k \in Nodes(j)} \frac{\sum_{s \in Services(k)} |RECC(j,k) - FEED(s,k)|}{|Services(k)|}$$

Now, to update the element $RREL_i(j)$, we use a weighted mean between the value of $RREL_i(j)$ at the previous step and the current recommendation reliability:

$$RREL_i(j) = \alpha \cdot RREL_i(j) + (1-\alpha) \cdot rr(k)$$

where α has the same meaning than for the case of the service reliability.

- **Phase 3: Computation of R and β.** The recommendations contained in $RECC_i$ are used by a_i to compute the reputations of the other nodes of the community. In particular, a_i computes the reputation of another node a_j as a weighted mean

of all the recommendations received by the other nodes concerning a_j, where the weight of each recommendation value is the recommendation reliability. Thus:

$$R_i(j) = \frac{\sum_{k \in AS, k \neq i} RECC_i(k,j) \cdot RR_i(k,j)}{\sum_{k \in AS, k \neq i} RR_i(k,j)}$$

The β coefficient associated to the agent a_i is recorded in the mapping β_i.

- **Phase 4: Computation of P.** The node a_i finally computes the overall preference measure $P_i(j)$ in the node a_j by considering both the service reliability $SR_i(j)$ and the reputation $R_i(j)$. In particular, the value of the mapping $\beta_i(j)$ is used to weight the importance of the service reliability with respect the reputation:

$$P_i(j) = \beta_i(j) \cdot SR_i(j) + (1 - \beta_i(j)) \cdot R_i(j)$$

At each step, the node a_i exploits the mapping P to select, among the nodes belonging to the admissible region $S(\overline{q})$, the most suitable candidates to require a collaboration.

4 Finding Suitable Nodes with SW-HYGRA

According to the schema reported in the Sections above, a node receiving a job request, which cannot directly fulfill, searches for another node (or a set of nodes) exposing an adequate amount of resources and, by exploiting reputation data, establishes whether it is best suited to perform the job. Since we are considering an environment model composed of a *huge* number of nodes[1], to support such a search process we adopt a *peer-to-peer* approach since it is known to be more efficient and scalable than a centralized solution [5, 11]. The schema adopted in this paper is derived from SW-HYGRA (*Small-World HYperspace-based Grid Resource Allocation*), a P2P resource finding approach for clouds/grids developed and studied by the authors in the recent years [6, 7]. SW-HYGRA is based on organizing the nodes in an *overlay network*, exploiting the links to surf the network, from node to node, until the one able to offer the required resource is found. The key aspect of SW-HYGRA is the algorithm adopted to construct the overlay network, which is also the basis for an efficient resource finding process. SW-HYGRA uses a geometric abstraction by modeling the entire system as a $n-$dimensional space where each *resource type* represents a *coordinate* whose value is the *available quantity* of the considered resource. Each node, on the basis of its resource availability, is represented as a *point* in the hyperspace, therefore its position changes each time a new job is allocated on it, or a running job terminates, freeing the resources no more used. A *metric* is introduced to measure the "distance" between two nodes, i.e. how much two nodes are "far" in terms of resource availability. To this aim, we exploited the Euclidean distance computed using node's coordinates.

[1] In an order which ranges from 10K to 1M.

4.1 Overlay Construction Algorithm

Overlay construction is the key for a fast and effective resource finding. It is performed by means of a decentralized algorithm which runs on each node of the network executing the following steps (say n^* a generic node):

1. node n^* contacts its linked/neighbor nodes in order to obtain, in turn, their linked nodes; this operation allows a node to obtain the set of linked nodes at *2-hops*;
2. the set is ordered by using the Euclidean distance of each node from n^*;
3. on the basis of some threshold parameters k and h, node n^* rearranges its links, interconnecting itself with at most k near nodes and at least h far nodes.

As reported in [6], which details algorithm performances obtained by means of some simulation measurements, the effect of the said steps is to create some *clusters of nodes* featuring a short intra-cluster distance, while keeping *long links* between clusters. Since the Euclidean distance is a measure of resource availability similarity, such clusters are characterized by nodes with a resource status very close to each other. By exploiting short links, a fast navigation inside the cluster is possible to e.g. refine a resource finding process, while, by using long links, it is possible to fast reach the region (i.e. the cluster) in the hyperspace where the nodes offering the requested resources reside. As proved in [7], the overlay network obtained features a structure quite similar to a *small-world* [12]; as it is known, such networks exhibit a high clustering degree and a very low average path length, characteristics which are very important to make resource finding effective. Some steps in the construction of an overlay network by exploiting the proposed algorithm are depicted in Figure 1.

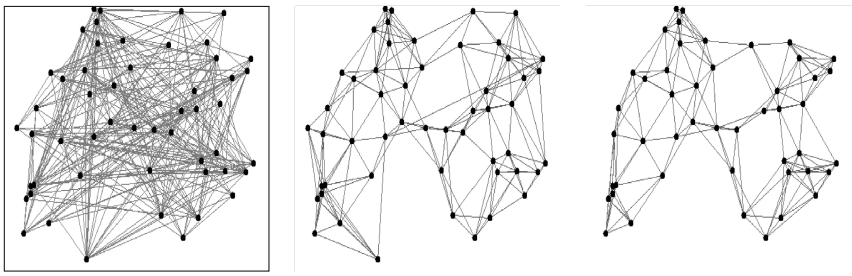


Fig. 1 The overlay network construction

4.2 Resource Finding

According to the said hyperspace abstraction, not only a node can be viewed as a point in the metric space but also a *resource request* can be represented in the same way. Indeed, requesting to execute a job implies to ask the system to allocate a certain amount of provided resources, such as *at least* a certain quantity of RAM, a certain number of CPU or cores, etc. Since such a request in general carries the

specific quantities of resources and, since resources are the coordinates of our hyperspace, the request denotes a *point* representing the lower-left corner of a *region* or *semi-space* whose internal nodes are those offering adequate resources for the given request. Such a semi-space is called in SW-HYGRA as the *admissible region* S, for the specific request and its discovery is the aim of the resource finding algorithm detailed below. As usual in P2P systems, also resource finding exploits a decentralized approach which is based on the following *check-and-forward* model:

1. A node receiving the request checks if it is able to fulfill it; if this is the case, the node belongs to the admissible region, so we reached the target and can continue with step 4;
2. if the node does not have sufficient resources, it contacts its neighbors and, on the basis of their resource status, forwards the request to one of them, selected using an appropriate heuristics; such an heuristics is chosen in order to help the request to reach the admissible region as soon as possible;
3. the algorithm keeps track of all the nodes visited (this set is carried together with the request), if all the possible nodes to jump onto are already analyzed, the system does not include a node suitable to host the request, so the algorithm terminates with failure;
4. when we found a node belonging to the admissible region, by suitably navigating through links the algorithm can reach other valid nodes, in order to build the set needed by the reputation schema.

A study on effectiveness, correctness and performances of this resource finding algorithm can be found in [6], as well as the evaluation of some forwarding heuristics. Results proved that, above all in presence of high overall system load (low resource availability), the algorithm described performs very well, ensuring to find the target node in less than 10 steps (on average) with a network size in the order of 10^5 nodes.

5 Experiments

In order to prove the effectiveness of the proposed approach, we performed a set of simulations with the same C-based simulation tool we used for the experimental analysis of the SW-Hygra system [6]. To this aim, we extended the basic test-bed by introducing the reputation model into the SW-HYGRA system, as explained below.

The role of *Task Allocator* has been defined in the following way: once it receives a request, it is forwarded to the most suitable node (see Section 2), which in turn will look for a collaboration with another node through the SW-HYGRA algorithm explained in Section 4.2. We modeled the capacity of the single node to provide a certain level of quality of service by the ratio $\frac{q^*}{q}$, where q^* is the actual amount of resources offered by the single node, and q is the amount of declared resources. Since in our experimental test-bed the coordinates of the nodes are based on the value of q (see Section 4.1), according to the above considerations, the nodes within the admissible region might not have enough resources to provide the maximum level of service, i.e. to fully satisfy the received request. We categorized the nodes into two subsets: T (with Trust model) includes the nodes which use the reputation

model whenever they have to select their collaborator; *WT* (Without Trust model) includes nodes which do not use the reputation model nor relative information.

The simulation has been performed in a test-bed of 10^5 nodes. We studied *i)* the effect of the different values of $\frac{q^*}{q}$ on the average QoS provided to the clients[2], and *ii)* the effect of the different values of the ratio $\frac{|WT|}{N}$, where N is the total number of nodes. The results, reported into Figures 2a and 2b, show that the integration of the competitive approach described into Sections 2 and 3 has makes a difference for the level of QoS provided by the whole system.

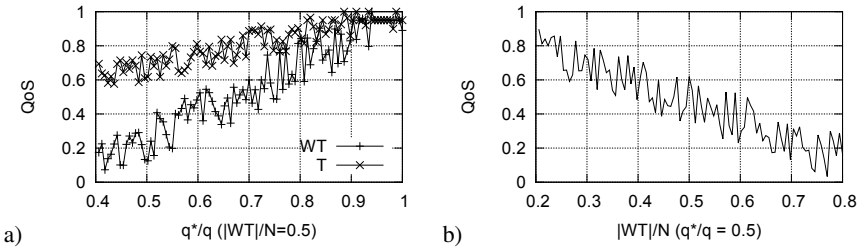


Fig. 2 Results of the: a) Average QoS for the sets WT and T; b) Average QoS vs ratio $\frac{|WT|}{N}$

Indeed, we observed that nodes in T provide, in average, a better level of satisfaction (QoS) than the others in WT (Figure 2a), and whenever the size of set WT grows over the 50% of the total, the QoS of the system drastically decreases (Figure 2b).

6 Conclusions

In this paper, we have presented a trust-based approach to support task allocation in a competitive cloud/grid system. Our approach, similarly to other similar techniques developed in the past for competitive agent systems, allows a node to choose the most promising interlocutors to obtain a collaboration, based on both a direct trust measure (reliability) and a reputation measure derived from the recommendations of the other nodes. However, differently from the aforementioned approaches, that generally explore the whole agent space for selecting the interlocutors, our technique exploits the SW-HYGRA Grid Resource Allocation for organizing the servers/computing nodes in an overlay network featuring certain characteristics. This way, our approach suitably makes the resource finding process efficient in a competitive cloud/grid system, since a node that has to find some interlocutor for a collaboration using its trust model can limit its search to an admissible region previously discovered by SW-HYGRA. The basic model of SW-HYGRA is to employ an overlay construction algorithm which exploits the resource status similarity, i.e. peers featuring a similar amount of resource availability tend to be interconnected by means

[2] It is collected in form of feedback according to Section 2.

of the links of the overlay thus forming clusters which, in turn, are connected together by means of few long links. Some experiments we have realized show that nodes using our trust-based approach perform significantly better than nodes that do not use any trust model to select their interlocutors. As for our ongoing research, we plan to extend our approach in order to consider nodes that dynamically change in time their resource capabilities. In this future scenario, the advantage of using a trust-based model should be even more significant.

References

1. Foster, I., Kesselman, C.: The grid: blueprint for a new computing infrastructure. Morgan Kaufmann (2004)
2. Furht, B., Escalante, A.: Handbook of Cloud Computing, 1st edn. Springer Publishing Company, Incorporated (2010)
3. Garruzzo, S., Rosaci, D.: The Roles of Reliability and Reputation in Competitive Multi Agent Systems. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 326–339. Springer, Heidelberg (2010)
4. Grandison, T., Sloman, M.: Trust Management Tools for Internet Applications. In: Nixon, P., Terzis, S. (eds.) iTrust 2003. LNCS, vol. 2692, pp. 91–107. Springer, Heidelberg (2003)
5. Iamnitchi, A., Foster, I.: A Peer-to-Peer Approach to Resource Location in Grid Environments. In: Grid Resource Management. Kluwer Pub. (2003)
6. Messina, F., Pappalardo, G., Santoro, C.: Decentralised Resource Finding in Cloud/Grid Computing Environments: a Performance Evaluation. In: Proc. of the 21th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises, Toulouse, France
7. Messina, F., Pappalardo, G., Santoro, C.: Exploiting the Small-World Effect for Resource Finding in P2P Grids/Clouds
8. Rosaci, D.: Trust measures for competitive agents. Know.-Based Syst. 28, 38–46 (2012)
9. Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. Artificial Intell. Review 24(1), 33–60 (2005)
10. Sabater-Mir, J., Paolucci, M.: On Representation and Aggregation of Social Evaluations in Computational Trust and Reputation Models. Int. J. Approx. Reasoning 46(3), 458–483 (2007)
11. Talia, D., Trunfio, P.: Toward a Synergy Between P2P and Grids. IEEE Internet Computing 7(4), 94–96 (2003)
12. Watts, D., Strogatz, S.J.: Collective Dynamics of 'Small-World' Networks. Nature 393(6684), 440–442 (1998)

Data Science and Distributed Intelligence: Recent Developments and Future Insights

Alfredo Cuzzocrea and Mohamed Medhat Gaber

Abstract. *Big Data*, *Data Science* and *MapReduce* are three keywords that have flooded our research papers and technical articles during the last two years. Also, due to the inherent distributed nature of computational infrastructures supporting Data Science (like *Clouds* and *Grids*), it is natural to view *Distributed Intelligence* as the most natural underlying paradigm for novel Data Science challenges. Following this major trend, in this paper we provide a background of these new terms, followed by a discussion of recent developments in the data mining and data warehousing areas in the light of aforementioned keywords. Finally, we provide our insights of the next stages in research and developments in this area.

1 Introduction

The two terms *Big Data* [28] and *MapReduce* [12] have dominated the scene in the intelligent data analysis field during the last two years. They are in fact the cause and effect of the rapid growth in data observed in the digital world. The phenomenon of very large databases and very high rate streaming data has been coined recently as Big Data. The largest two databases for Amazon account for 42 terabytes of data in total, and YouTube receives at least 65,000 new videos per day. Such figures increase every day and we are literally drowning in high waves of data. Making sense out of this data has become more important than ever in the knowledge era. With the birth of *learning from data streams*, Mutukrishnan in his later published book [24] has defined data streams as "*data arriving in a high rate that challenges our computation and communication capabilities*". In fact, this definition is now more true than back then. In spite of the continuous advances in our computation and communication capabilities, the data growth has been much faster, and the problem has become even more challenging. As a natural reaction to this worsening, a number of

Alfredo Cuzzocrea · Mohamed Medhat Gaber
ICAR-CNR and University of Calabria, I-87036 Cosenza, Italy
School of Computing, University of Portsmouth, Portsmouth PO1 3HE, Hampshire, UK

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 139–147.
springerlink.com

advanced techniques for data streams have been proposed, ranging from *compression paradigms* (e.g., [10, 11]) to intelligent approaches that successfully exploit the nature of such data sources, like their *multidimensionality*, to gain in effectiveness and efficiency during the processing phases (e.g., [8]), and recent initiatives that are capable of dealing with complex characteristics of such data sources, like their *uncertainty* and *imprecision*, as dictated by modern stream applicative settings (e.g., *social networks*, *Sensor Web*, *Clouds* – [9]).

Addressing such challenges has kept Data Mining and Machine Learning practitioners and researchers busy with exploring the possible solutions. MapReduce has come as a potentially effective solution when dealing with large datasets, by enabling the breakdown of the main process into smaller tasks. Each of these tasks could be performed either in a *parallel* or *distributed* processing mode of operation. This allows the speed-up of performing complex data processing tasks, in an attempt to catch up with high speed large volume of data generated by scientific applications, such as the promising contexts of *analytics over large-scale multidimensional data* and *large-scale sensor network data processing*. With Big Data and MapReduce at the front of the scene, a new term describing the process of dealing with very large dataset has been coined, *Data Science*.

In line with this, when these kind of dataset are processed on top of a service-oriented infrastructure like the novel *Cloud Computing* one [2], the terms "*Database as a Service*" (DaaS) [19] and "*Infrastructure as a Service*" (IaaS) arise, and it is become critical to understand how Data Science can be coupled with distributed, service-oriented infrastructures, with novel and promising computational metaphors. Hence, due to the inherent distributed nature of computational infrastructures like Clouds (but also *Grids* [15]), it is natural to view *Distributed Intelligence* as the most natural underlying paradigm for novel Data Science challenges.

Following this major trend, in this paper we highlight the development in the Data Science area by first providing the necessary background of the Big Data and MapReduce in Section 2. Recent developments in the Data Mining field with the emergence of data science are provided in Section 3, followed by the recent developments on OLAP and Data Warehousing in Section 4. In Section 5, we present the foreseen future development in this area. Finally, in Section 6, we provide a summary and conclusions of our research.

2 The Emergence of Data Science

In his famous article "*What is Data Science?*" [23], Loukides has enumerated differences between Data Science and traditional statistical analysis. Mainly, Data Science deals with the whole process of gathering data, pre-processing them and finally making sense out of them, producing what he termed as *data products*. This definition may be confused with any definition given to Data Mining and Data Warehousing processes. What really makes Data Science different, however, is the holistic approach when looking at producing a data product. This is especially true with the large volumes of noisy and unstructured data generated in our daily lives,

from social media to search terms on Google. Traditional Data Mining and Warehousing strategies become no longer valid when dealing with such large and dynamic data sources.

Thus, the phenomenon of Big Data has dictated the emergence of a new field that encompasses a number of well-established areas, including at the front line, Data Mining and Warehousing. This is the Data Science field, a term that we will encounter very often for some years to come. Scaling up the data analysis techniques to cope with Big Data has spotted the light on old functional programming functions, `map` and `reduce`, giving raise to the MapReduce computational paradigm. In the following subsections, a discussion of the Big Data phenomenon and how the two functions `map` and `reduce` have helped scaling up Big Data problems within the MapReduce paradigm is provided.

2.1 The Big Data Phenomenon

Soulellis [27] has enumerated a number of examples of Big Data. These include: (*i*) approximately, one zettabyte (i.e., $1,000,000,000,000$ bytes) of data have been produced in 2010; (*ii*) it is estimated that 8 zettabytes of data will be produced in 2015; (*iii*) more than $30,000$ tweets are sent every minute actually. All these examples well-describe the Big Data phenomenon that characterizes actual information systems. More interestingly and in addition to these examples, 90% of our data was the result of only the last two years of data production.

As a consequence, we are facing a big challenge with such huge data, and adequate analysis of these data can help advancing our knowledge greatly. There is no doubt that there is a great business advantage when enterprises are able to use such data to guide their decision making. It is a well-known news story that *GAP* store chain management have reverted their decision to change the company's logo when sentiment extracted from social media revealed that the customers did not like the new logo [4]. Another example is the controversial news story that *TARGET* department store have been able to predict that a teenage girl is pregnant using her new pattern of purchasing [21].

Not only business enterprises can benefit from such large data repositories, scientific discoveries can be also drawn from big data collected using advanced instruments generating data at very high rates. *Galaxy Zoo* [22] is one example of large data repository that uses the emergence of citizen science. Citizen science uses crowd annotation and data collection for the use in scientific research. In *Galaxy Zoo*, a very large collection of images representing galaxies are provided for users to annotate.

2.2 The MapReduce Computational Paradigm

MapReduce is a programming model that uses a *divide and conquer* method to speed-up processing large datasets [12]. It has been used in 2003 for the implementation of inverted index within the *Google Search Engine* in order to efficiently

handle the search process. Also, it has been successfully exploited to handle large scale Machine Learning and Text Analytics tasks within *Google Analytics*. *Hadoop* [3] is the widely-known open source implementation of MapReduce.

The model of *MapReduce* has two main functions (`map` and `reduce`). The `map` function processes a key/value pair to produce a number of intermediate key/value pairs, which are then processed using the `reduce` function to merge all the intermediate pairs, and obtain the final result. A simple example by Dean and Ghemawat [12] has a `map` function that takes a string, and outputs the value 1 for each occurrence of a word in that string. The `reduce` function in turn processes this output to sum up the total occurrences for each word in the given string. MapReduce has attracted a great deal of attention over the past five years. More than 100,000 jobs uses MapReduce run on Google clusters every day [12]. Similar examples could be found in other giant software firms. MapReduce is a natural way for distributed and parallel processing of large datasets. Thus, Big Data has found a feasible way to be consumed by processing applications. However, the pace of data generation is still a big issue. This is especially true when we deal with Data Mining and Warehousing applications.

MapReduce is strictly related to the DaaS and IaaS metaphors mentioned in Section 1. Here, we discuss DaaS and IaaS in greater detail. DaaS defines a set of tools that provide final users with seamless mechanisms for creating, storing, accessing and managing their proper databases on remote (data) servers. Due to the naive features of Big Data, DaaS is the most appropriate computational data framework to implement Big Data repositories [2]. MapReduce is a relevant realization of the DaaS initiative. IaaS is a provision model according to which organizations outsource infrastructures (i.e., hardware, software, network) used to support ICT operations. The IaaS provider is responsible for housing, running and maintaining these services, by ensuring important capabilities like elasticity, pay-per-use, transfer of risk and low time to market. Due to specific application requirements of applications running over Big Data repositories, IaaS is the most appropriate computational service framework to implement Big Data applications.

3 Recent Data Mining Developments

A number of Data Mining techniques have been developed utilizing the MapReduce framework to scale up to Big Data. Papadimitriou *et al* [26] have classified the applicability of speedingup Data Mining algorithms using MapReduce into the following three categories: *one-iteration techniques* that are perfect for MapReduce, *multiple-iterations techniques* that are applicable taking into consideration that only small amount of shared information needs to be synchronized among iterations, and *not- applicable techniques* that typically require large shared information to be synchronized. Examples of the first category are *Canopy* for clustering and *Naive Bayes* and *K-Nearest Neighbours* (K-NN) for classification. *K-means* for clustering and *Gaussian Mixture* for classification represent typical examples of the second category. Finally, *Support Vector Machine* (SVM) cannot easily utilize MapReduce

for speeding-up the execution of the method, hence it is a well representative of the third category. However, it is still possible to design an implement techniques falling in the third category in a way that they can utilize the benefits of using the MapReduce framework.

In clustering, Ene *et al* [14] have developed two clustering algorithms, namely, *MapReduce-kCenter* and *MapReduce-kMedian*, targeted to extend classical Data Mining methods to MapReduce framework, for efficiency purposes. The two algorithms run in a constant number of MapReduce rounds achieving a constant factor approximation. Experimentally, significant speed up of the proposed techniques have been reported. Another clustering technique has been proposed by Corderio *et al* in [7]. This technique aims at minimizing the I/O and the network cost, proposing the so-called "*Best of both Worlds*" *BoW* technique, which supports subspace clustering on very-large high-dimensional datasets on top of MapReduce. Papadimitriou and Sun [25] have used the MapReduce framework to develop a distributed co-clustering algorithm, that has been coined *DisCo*. Experimentally the technique was able to scale up to several hundreds to gigabytes of data. Utilization of MapReduce for K-means has been reported in [5], proving that a speed up of an average of 1.937 can be achieved on a dual core processor. A more generic contribution has been developed by Ghoting *et al* [17], which propose a generic toolkit for the development of Data Mining algorithms using MapReduce, termed as *NIMBLE*. In classification, Chu *et al* [5] have also utilized MapReduce to speed-up Naive Bayes, *Neural Networks*, *Logistic Regression* and *Linear SVM*. It has been experimentally proven that running these techniques on a dual core processor speeds-up target techniques approximately by 2 times (in average: 1.950 for Naive Bayes, 1.905 for Neural Networks, 1.930 for Logistic Regression, and 1.819 for Linear SVM).

4 Recent Data Warehousing and OLAP Developments

Among the recent advances on Data Warehousing and OLAP over Big Data, *analytics over Big Data* play a relevant role in this respect. Let us focus on this research challenge in a greater detail. Analytics can be intended as complex procedures running over large-scale, enormous-in-size data repositories (like Big Data repositories) whose main goal is that of extracting useful knowledge kept in such repositories. Two main problems arise, in this respect. The first one is represented by the issue of conveying Big Data stored in heterogeneous and different-in-nature data sources (e.g., legacy systems, Web, scientific data repositories, sensor and stream databases, social networks) into a structured, hence well-interpretable, format. The second one is represented by the issue of managing, processing and transforming so-extracted structured data repositories in order to derive *Business Intelligence* (BI) components like diagrams, plots, dashboards, and so forth, for decision making purposes. Actually, both these aspects are of emerging interest for a wide spectrum of research communities, and more properly for the Data Warehousing and OLAP research community. As a consequence, this has generated a rich literature. At the industrial research side, Hadoop [3] and *Hive* [29] are two fortunate implementations of

the ETL (*Extraction-Transformation-Loading*) layer and the BI layer of Big Data applications, respectively.

Although analytics over large-scale data repositories have been deeply investigated recently, the problem of extending actual models and algorithms proposed in this respect to the specific *Big Multidimensional Data* context plays a leading role, as multidimensional data naturally marry with analytics. Analytics over Big Data repositories has recently received a great deal of attention from the research communities. One of the most significant application scenarios where Big Data arise is, without doubt, scientific computing. Here, scientists and researchers produce huge amounts of data per-day via experiments (e.g., think of disciplines like high-energy physics, astronomy, biology, bio-medicine, and so forth) but extracting useful knowledge for decision making purposes from these massive, large-scale data repositories is almost impossible for actual DBMS-inspired analysis tools.

In response to this "*computational emergency*", the Hadoop system has been proposed, as above-mentioned. Hadoop runs MapReduce tasks over Big Data, and also it makes available the *Hadoop Distributed File System* (HDFS) [3] for supporting file-oriented, distributed data management operations efficiently. It has been highlighted that Hadoop is a kind of *MAD* system [6] meaning that (*i*) it is capable of attracting all data sources (*M* standing for *Magnetism*), (*ii*) it is capable of adapting its engines to evolutions that may occur in big data sources (*A* standing for *Agility*), (*iii*) it is capable of supporting depth analytics over big data sources much more beyond the possibilities of traditional SQL-based analysis tools (*D* standing for *Depth*). In a sense, Hadoop can be reasonably considered as the evolution of next-generation Data Warehousing systems, with particular regards to the ETL phase of such systems.

Several studies, like [13], have provided recommendations for further improving the computational capabilities of Hadoop, whereas [1] proposes *HadoopDB*, a novel hybrid architecture that combines MapReduce and traditional DBMS technologies for supporting advanced analytics over large-scale data repositories. Furthermore, *Starfish* [20] is a recent self-tuning system for supporting big data analytics that is still based on Hadoop but it incorporates special features trying to achieve higher performance by means of *adaptive metaphors*. By looking at BI aspects of analytics over big data, the state-of-the-art research result is represented by Hive [29], a BI system/tool for querying and managing structured data built on top of the Hadoop's HDFS. Hive which allows us to obtain the final analytics components (in the form of diagrams, plots, dashboards, and so forth) from the big data processed, materialized and stored via Hadoop. Also, Hive introduces a SQL-like query language, called *HiveQL* [29], which runs MapReduce jobs immersed into SQL statements.

5 What Is Next?!

Web 2.0 applications will continue generating Big Data for few years to come, along with the ever increasing volumes of scientific data that is generated continuously and in a very high data rate. Smartphones as an emerging source of Big

Data have started to provide us with rich source of fine-grained sensory data. The data gravity principle has never been as true as today and it will become even more important in the near future. This principle as stated by the Data Mining guru *Gregory Piatetsky-Shapiro* is that *"the bigger the data, the harder it is to move it, so logic need to come to big data"*. Some new directions that are likely to continue in the Data Mining research area related to Big Data include the following topics. (*i*) *Mobile Data Mining* A focus of performing Data Mining locally on handheld devices has attracted a great deal of attention recently. Addressing the issues of limited resources and changing context of mobile users has been addressed in a large number of proposals. Examples include the work by Gaber [16] and Gomes *et al* [18]. (*ii*) *Embedded Data Mining in Wireless Sensor Networks* It has been proved experimentally that in-network processing of wireless sensor networks is the most feasible mode of operation for such networks. Accordingly, a number of techniques have been developed to mine data on board wireless sensor nodes. Accordingly, a number of techniques for mining data on board wireless sensor nodes have been developed (e.g., [30]).

As regards to Data Warehousing and OLAP research area related to Big Data, there still are a number of open research problems, some of which can be summarized by the following questions. (*i*) *How To Directly Integrate Multidimensional Data Sources Into The Hadoop Lifecycle?* Hadoop populates the underlying structured Big Data repositories from heterogeneous and different-in-nature data sources, such as legacy systems, Web, scientific data sets, sensor and stream databases, social networks, and so forth. Despite this, no research efforts have been devoted to the yet-relevant issue of *directly integrating multidimensional data sources* into the Hadoop lifecycle, which is an exciting research challenge for next-generation Data Warehousing and OLAP research. (*ii*) *How To Model and Design Multidimensional Extensions of HiveQL?* In order to achieve an effective integration of multidimensional data models with analytics over Big Data, the query language HiveQL must be enriched with multidimensional extensions as well. These extensions should take into consideration language syntax aspects as well as query optimization and evaluation aspects, perhaps by inheriting lessons learned in the context of actual *MDX-like languages* for multidimensional data. (*iii*) *How to Design Complex Analytics over Hadoop-Integrated Multidimensional Data?* Multidimensional data provide add-on value to Big Data analytics. In this respect, design complex analytics over Hadoop-integrated multidimensional data plays a critical role. Actual analytics, although quite well-developed, still do not go beyond classical BI components, like diagrams, plots, dashboards, and so forth, but complex BI processes of very large organizations demand for *more advanced BI-oriented decision support tools*, perhaps by integrating principles and results of different-in-nature disciplines like statistics.

6 Summary and Conclusions

In this paper, we have discussed the emergence of Data Science and its consequent developments in the areas of Data Mining and Data Warehousing. We have also put

in emphasis the natural marriage between Data Science and Distributed Intelligence paradigms, due to the inherent distributed nature of computational infrastructures supporting Data Science (like Clouds and Grids). As active researchers in this field, we have also highlighted possible future directions for further developments of both Data Mining and Data Warehousing areas related to Big Data. We observe how these directions will result from the Big Data phenomenon with extreme high gravity distribution. New models of data processing will be required in the near future, opening the door for new key players to take leading roles in the market.

References

1. Abouzeid, A., et al.: Hadoopdb: An architectural hybrid of mapreduce and dbms technologies for analytical workloads. PVLDB 2(1), 922–933 (2009)
2. Agrawal, D., Das, S., Abbadi, A.E.: Big data and cloud computing: current state and future opportunities. In: EDBT, pp. 530–533 (2011)
3. Apache. Hadoop (July 2011), http://wiki.apache.org/hadoop
4. Gap, B.: scraps new logo after online outcry (2010), http://www.bbc.co.uk/news/business-11520930
5. Chu, C.-T., et al.: Map-reduce for machine learning on multicore. In: NIPS, pp. 281–288 (2006)
6. Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J.M., Welton, C.: Mad skills: New analysis practices for big data. PVLDB 2(2), 1481–1492 (2009)
7. Cordeiro, R.L.F., et al.: Clustering very large multi-dimensional datasets with mapreduce. In: KDD, pp. 690–698 (2011)
8. Cuzzocrea, A.: CAMS: OLAPing Multidimensional Data Streams Efficiently. In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 2009. LNCS, vol. 5691, pp. 48–62. Springer, Heidelberg (2009)
9. Cuzzocrea, A.: Retrieving Accurate Estimates to OLAP Queries over Uncertain and Imprecise Multidimensional Data Streams. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 575–576. Springer, Heidelberg (2011)
10. Cuzzocrea, A., Chakravarthy, S.: Event-based lossy compression for effective and efficient olap over data streams. Data Knowl. Eng. 69(7), 678–708 (2010)
11. Cuzzocrea, A., Furfaro, F., Mazzeo, G.M., Saccá, D.: A grid framework for approximate aggregate query answering on summarized sensor network readings. In: OTM Workshops, pp. 144–153 (2004)
12. Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Commun. ACM 51(1), 107–113 (2008)
13. Dittrich, J., Quiané-Ruiz, J.-A., Jindal, A., Kargin, Y., Setty, V., Schad, J.: Hadoop++: Making a yellow elephant run like a cheetah (without it even noticing). PVLDB 3(1), 518–529 (2010)
14. Ene, A., Im, S., Moseley, B.: Fast clustering using mapreduce. In: KDD, pp. 681–689 (2011)
15. Foster, I.T., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. IJHPCA 15(3), 200–222 (2001)
16. Gaber, M.M.: Data stream mining using granularity-based approach. In: Foundations of Computational Intelligence, vol. (6), pp. 47–66. Springer (2009)

17. Ghoting, A., Kambadur, P., Pednault, E.P.D., Kannan, R.: Nimble: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In: KDD, pp. 334–342 (2011)
18. Bártolo Gomes, J., Gaber, M.M., Sousa, P.A.C., Menasalvas, E.: Context-Aware Collaborative Data Stream Mining in Ubiquitous Devices. In: Gama, J., Bradley, E., Hollmén, J. (eds.) IDA 2011. LNCS, vol. 7014, pp. 22–33. Springer, Heidelberg (2011)
19. Hacigümüs, H., Mehrotra, S., Iyer, B.R.: Providing database as a service. In: ICDE, pp. 29–38 (2002)
20. Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F.B., Babu, S.: Starfish: A self-tuning system for big data analytics. In: CIDR, pp. 261–272 (2011)
21. Hill, K.: How target figured out a teen girl was pregnant before her father did. Forbes (2012)
22. Lintott, C.J., Schawinski, K., Slosar, A., Land, K., Bamford, S., Thomas, D., Raddick, M.J., Nichol, R.C., Szalay, A., Andreescu, D., Murray, P., Vandenberg, J.: Galaxy Zoo: morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey. Monthly Notices of the Royal Astronomical Society 389(3), 1179–1189 (2008)
23. Loukides, M.: What is data science? the future belongs to the companies and people that turn data into products. An OReilly Radar Report (June 2010)
24. Muthukrishnan, S.: Data streams: algorithms and applications. Foundations and trends in theoretical computer science. Now Publishers (2005)
25. Papadimitriou, S., Sun, J.: Disco: Distributed co-clustering with map-reduce: A case study towards petabyte-scale end-to-end mining. In: ICDM, pp. 512–521 (2008)
26. Papadimitriou, S., Sun, J., Yan, R.: Large-scale data mining: Mapreduce and beyond. In: Tutorial in KDD (July 2010)
27. Soulellis, G.: Emerging trends in big data and analytics. Big Data Innovation, London (2012)
28. Stonebraker, M., Hong, J.: Researchers' big data crisis; understanding design and functionality. Commun. ACM 55(2), 10–11 (2012)
29. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Anthony, S., Liu, H., Murthy, R.: Hive - a petabyte scale data warehouse using hadoop. In: ICDE, pp. 996–1005 (2010)
30. Yin, J., Gaber, M.M.: Clustering distributed time series in sensor networks. In: ICDM, pp. 678–687 (2008)

Data-Flow Awareness in Parallel Data Processing

David Bednárek, Jiří Dokulil, Jakub Yaghob, and Filip Zavoral

Abstract. The memory hierarchy affects the performance of task-scheduling strategies in task-based parallel environments. For data-intensive problems, the flow of data may be explicitly specified as a part of the algorithm, allowing the task scheduler to be aware of the data flow. In this paper, we describe such a task-based environment with explicit data-flow specification. We demonstrate the effect of data-flow awareness on the system performance. The results show that the explicit specification of data flow improves the quality of task scheduling.

1 Introduction

One of the approaches of parallel software development is *task parallelism* where the job is divided into a set of *tasks*. A *scheduler* is responsible for the selection of the next task that will be run in the same thread once the previous task is done. The scheduling strategy may affect the overall performance in many ways; e.g., the performance is sensitive to the effect of memory hierarchy (using caches). Unfortunately, schedulers in contemporary parallelization systems (TBB [1], OpenMP [2], or MPI [3]) do not have enough information on data flow, so it is difficult to optimize their scheduling strategy with respect to the memory hierarchy.

We focus on the area of data-intensive scientific computing where the cost of data access forms a significant portion of the overall cost; on the other hand, the computation aspect of the problem is typically too difficult to be divided into a set of completely independent tasks. Consequently, the communication between tasks significantly affects the performance. We show that a task-baed system may significantly benefit from explicit specification of data-flow information. We have used the Bobox parallelization framework [4] which was developed for scientific experiments in databases, streams and data processing in general [5–7].

David Bednárek · Jiří Dokulil · Jakub Yaghob · Filip Zavoral
Charles University Prague, Czech Rep.
e-mail: {bednarek,dokulil,yaghob,zavoral}@ksi.mff.cuni.cz

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 149–154.
springerlink.com

2 Parallel Execution Environment

The data flow is specified by a user in a form of a *Bobox plan* that contain *boxes* (computation units) connected to a nonlinear pipeline. One box processes tasks in series; boxes communicate using *envelopes* that contain data of the tasks. The role of the scheduler is dynamically assigning the tasks to the processors. Instead of having a central task scheduler, we propose running a separate scheduler for each thread in the pool that maintains two queues of tasks. *Immediate queue* is dedicated for tasks, which should be immediately scheduled while the data are hot in cache, *stealing queue* holds tasks, which are not tightly bound to the thread. When a task finishes, the scheduler starts the execution of the first task in the immediate queue. If the queue is empty, the scheduler tries to schedule the head of the stealing queue. If the stealing queue is empty, the scheduler *steals* tasks from another scheduler.

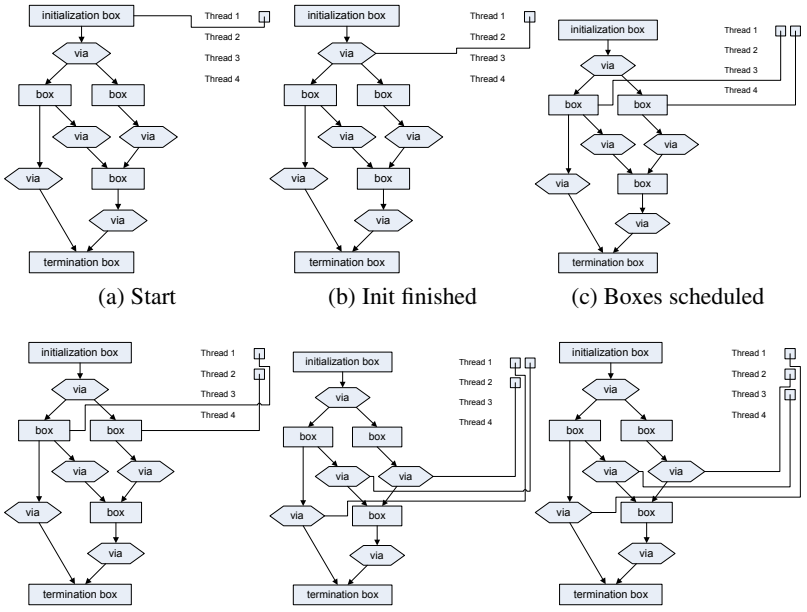


Fig. 1 Scheduling example

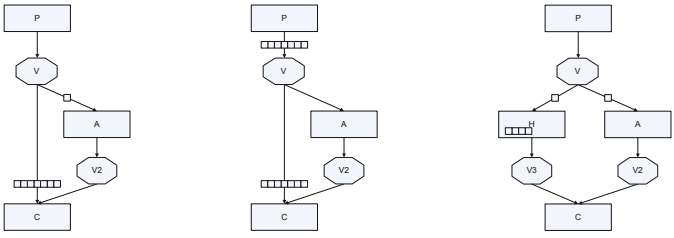
A task can be enqueued either as an *immediate task* or as a *relaxed task*. An immediate task is placed at the head of the immediate queue, whereas a relaxed task is placed at the end of the stealing queue. The selection of enqueueing is hard-wired into a framework. The main reason is the cache-awareness; since the head of the immediate queue should be scheduled immediately by the same thread on the same CPU, the data bound to the task is likely hot in the cache. A relaxed task is placed on the tail since the data are not needed to be hot.

An example of the scheduling of four theads is displayed in Fig. 1. First, the initial box is enqueued. When it is invoked, it produces a special marker (*poisoned pill*), sends it to the first via and enqueues the via (1b). The via duplicates the poisoned pill, sends it to the boxes and enqueues them (1c). The thread no. 2 steals one of the tasks (1d) and both tasks are invoked, the boxes produce the results and send them to the vias. The newly created tasks are enqueued with the same thread (1e). In the last step (1f), one of the task gets stolen by the thread no. 3.

3 Flow Control

Consider a part of a pipeline that consist of two boxes. *P* produces data and sends it to *C* that performs a complex computation. At some time after the start of the computation, *P* generates the first data, sends it to *V* and enqueues itself. The via *V* that connects *P* to *C* receives the first envelope and is also enqueued. After invocation of *P* ends, *V* is immediately invoked by the scheduler. *V* forwards the envelope to *C*, enqueues *C* and ends the invocation. *C* is then invoked and starts its long computation. At some point, the task corresponding to *P* gets stolen by another scheduler and invoked. Another envelope is produced and sent to *C*.

If this went on long enough, a great number of envelopes would be created and remain waiting. This could consume a lot of memory – up to the total size of data produced by *P*. To prevent wasting the memory, the size of the input buffer of each box is limited. If the buffer is full (*congested*), further data is rejected. When data is removed from the buffer, the via that sends the data is notified and resends them. This way, the congestion can propagate in the direction opposite to the flow of data.



(a) Initial state (b) Congestion starts (c) Helper box

Fig. 2 Congestion example – compute average

Since the pipeline may be nonlinear, there may exist deadlock situations. Consider a stream of numbers to filter out numbers that are above the average. The pipeline could split the data into two branches – this is done by a via *V* with one input and two outputs (Fig. 2). One branch does nothing with the data and the other computes the average. These branches are then combined by a box *C* – the data

stream and the average value. This box cannot process the data stream until it receives the average. But, if the number of envelopes in the data stream is larger than the total size of buffers, the box *C* would cause congestion which will propagate up to the via *V* and the via *V* will also start rejecting any further inputs (Fig. 2b).

The solution is left to the model generation module (e.g. SPARQL compiler [5]). It has to identify such situations and solve it by adding an extra helper box *H* (Fig. 2c) to the pass-through branch. The new box should accept all incoming envelopes. This way, the *C* box does not get congested and the *A* box can get the complete data and compute the average value. It means that all of the data is kept in the memory, but this cannot be avoided in this scenario. However, there are alternative solutions. For instance, if the stream can be computed twice with reasonable cost, then both branches can have their own source and even if the pass-through branch gets congested, it does not create the deadlock.

4 Experiments

4.1 DFA Scheduler

In order to evaluate the impact of the data-flow awareness, we performed series of tests using data-flow aware (DFA) and referential (non data-flow aware) schedulers. The test simulates a stream processing application where a complicated floating point operation needs to be performed on each item. The pipeline consisted of 190 boxes, each of which performed four trigonometric functions on each item.

The results (see Table 1, Graph 3) show the times (in millisec. per a megabyte of data) that were achieved with different envelope sizes. The optimum is affected by two main aspects – scheduling granularity and cache size.

Table 1 Execution times of two schedulers (in ms)

	2K	4K	8K	16K	32K	64K	128K	256K
DFA	1686	1267	1046	947	911	977	1420	2935
ref	1927	1423	1313	1297	1751	3002	3007	4044

The scheduling granularity works in two opposite directions. Larger envelope size means that tasks take longer to complete and their number is reduced; this reduces the scheduling overhead. On the other hand, it reduces the amount of time the system is able to run in parallel. The main effect of the cache is that if the data become larger than the available cache, the performance decreases. In simple setups, a single point where the data no longer fit into a cache can usually be identified as a sudden drop in performance. Since most CPUs have several levels of caches of increasing sizes, several drops are usually present.

The results of the DFA scheduler provide significantly better performance regardless of the data size. Moreover, the best results of the DFA scheduler is wider than in non-DFA scheduler; its performance tuning is much easier. The results of

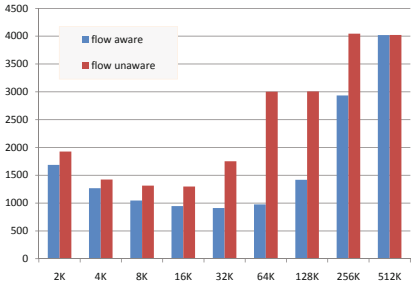


Fig. 3 Results of the experiments (time in ms / size of envelopes)

the non-DFA scheduler show high sensitivity to the cache sizes (64 KB L1 cache, 256 KB L2 cache); there is a performance degradation where the data exceeds the cache size. When the size of the data chunk exceeds the size of the cache (512 KB), the advantage of the DFA scheduler is eliminated, because each row is forced out of the cache and all operations are handled by the main memory.

4.2 Parallelism

To test speedup of a single execution and limit the influence of other factors, we created an "ideal conditions" test. We use a linear pipeline with 12 boxes. The first box produces a stream of data, then there are ten boxes that perform computation intensive operation on those data. The 12-th box receives the data. Table 2 shows the results of the test performed on 256 envelopes for different number of threads.

Table 2 Query performance for different number of threads

# of threads	time (ms)	std. dev.	speedup	# of threads	time (ms)	std. dev.	speedup
1	13416	8.4	1	8	3358	607.1	3.99
2	6741	11.0	1.99	16	9391	2519.6	1.43
4	3708	188.3	3.62	32	17104	3044.1	0.78

The results are close to what was expected – for 2 and 4 threads, there is a significant speedup over 1 thread. When 8 threads are used, the speedup is lower (the CPU has 4 physical cores and the 8 logical cores are provided by HyperThreading). The actions are computation-intensive, thus reducing the effect of HyperThreading. With higher number of threads than logical cores, the scheduling becomes impaired by preemptive multitasking actions. The dramatic fall in performance also suggests that the application-level scheduler is significantly more effective than the operating system scheduler for this type of problems.

5 Conclusions

We have demonstrated the impact of supplying data-flow information to a scheduler. From this point of view, the architecture of Bobox has the following advantages: (1) data-flow information is automatically a part of the application code, (2) the time required by the data-flow awareness in the scheduler is negligible, (3) the scheduler is able to cooperate with the application in the selection of the optimal task size, and (4) the improvement of the overall performance is significant.

The experiments have shown that data-awareness provides a significant performance increase. When compared to the optimal set-up of the non-DFA scheduler, the optimal setup of DFA scheduler required 30% less time; when both schedulers work with the same chunk size, the DFA scheduler always outperforms the non-DFA scheduler, unless the chunk size exceeds the cache size, in which case the cache is rendered useless in our test.

Many parallelization libraries offer the pipeline paradigm; however, linear pipeline is insufficient for many applications and their developers must use generic task parallelism which does not explicitly specify the data flow. As this paper shows, the data-flow awareness is beneficial and the ability of Bobox system to maintain non-linear pipeline is its important advantage.

Acknowledgements. This work was supported by the Grant Agency of the Czech Republic, grant number 202/10/0761 and SVV-2012-2653122.

References

1. Kukanov, A., Voss, M.J.: The foundations for scalable multi-core software in Intel Threading Building Blocks. Intel. Technology Journal 11(04), 309–322 (2007)
2. OpenMP Application Program Interface, V. 3.0, OpenMP Architecture Review Board (May 2008), http://www.openmp.org/mp-documents/spec30.pdf
3. MPI: A Message-Passing Interface Standard, Version 2.2, Message Passing Interface Forum (September 2009), http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf
4. Bednarek, D., Dokulil, J., Yaghob, J.: The Bobox Project - Parallelization Framework and Server for Data Processing, Charles University in Prague, Tech. Rep. 2011/01 (2011)
5. Cermak, M., Falt, Z., Dokulil, J., Zavoral, F.: SPARQL Query Processing Using Bobox Framework. In: The 5th Int. Conf. on Advances in Semantic Processing, SEMAPRO. Xpert Publishing Services, Sliema (2011)
6. Bednarek, D., Dokulil, J., Yaghob, J., Zavoral, F.: Using Methods of Parallel Semi-structured Data Processing for Semantic Web. In: 3rd International Conference on Advances in Semantic Processing, pp. 44–49. Xpert Publishing Services, Sliema (2009)
7. Falt, Z., Bednarek, D., Cermak, M., Zavoral, F.: On Parallel Evaluation of SPARQL Queries. In: The Fourth International Conference on Advances in Databases, Knowledge, and Data Applications DBKDA 2012. Xpert Publishing Services (2012)

Efficient Group Communication for Large-Scale Parallel Clustering

David Pettinger and Giuseppe Di Fatta

Abstract. Global communication requirements and load imbalance of some parallel data mining algorithms are the major obstacles to exploit the computational power of large-scale systems. This work investigates how non-uniform data distributions can be exploited to remove the global communication requirement and to reduce the communication cost in iterative parallel data mining algorithms. In particular, the analysis focuses on one of the most influential and popular data mining methods, the k-means algorithm for cluster analysis. The straightforward parallel formulation of the k-means algorithm requires a global reduction operation at each iteration step, which hinders its scalability. This work studies a different parallel formulation of the algorithm where the requirement of global communication can be relaxed while still providing the exact solution of the centralised k-means algorithm. The proposed approach exploits a non-uniform data distribution which can be either found in real world distributed applications or can be induced by means of multi-dimensional binary search trees. The approach can also be extended to accommodate an approximation error which allows a further reduction of the communication costs.

1 Introduction

In spite of the large number of clustering algorithms available in the litterature [1], the longevity and popularity of the k-means algorithm [2] [3] is witnessed by its perpetual adoption in many applicative domains.

Given a set of N input patterns and a user-defined parameter K, k-means determines a set of K points, called centres or centroids, so as to minimise the mean-squared distance from each data point to its nearest centre. The k-means algorithm is an iterative refinement process, where at each iteration the clustering assignments are updated, consequently changing the definition of the clusters.

David Pettinger · Giuseppe Di Fatta
School of Systems Engineering, The University of Reading, Whiteknights, Reading, Berkshire, RG6 6AY, UK
e-mail: {D.G.Pettinger,G.DiFatta}@reading.ac.uk

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 155–164.
springerlink.com

Multi-dimensional binary search trees (KD-Trees) can be used for very efficient nearest neighbour searches. KD-Trees and, in general, partitional trees, have been adopted to improve the efficiency of the sequential k-means algorithm [4, 5].

While a straighforward parallel formulation [6] of the k-means algorithm has been widely adopted, parallel k-means algorithms based on KD-Trees have not, as they are expected to have an irregular distribution of the computation load and to suffer from load imbalance [7, 8]. While this issue has so far prevented the adoption of these approaches in parallel computing environments, here we show that it can be turned into a main advantage.

Another important limitation of parallel k-means algorithms is that their communication costs and synchronisation requirements do not scale well in the number of processes. At each k-means iteration a global all-to-all reduction operation is required. Obviosly this is preventing their adoption in very large-scale parallel and distributed systems.

This work proposes a communication-efficient parallel formulation for the k-means algorithm based on KD-Trees for large and extreme-scale parallel systems. The combination of a data partitiong scheme and a novel efficient operation for group communication shows better scalability than the straightforward parallel k-means approach with respect to the communication costs.

The rest of the paper is organised as follows. Section 2 recalls the basic k-means algorithm. Section 3 reviews parallel k-means formulations and the use of KD-Trees for speeding up the convergence of the iterative process. Section 4 introduces a communication-efficient group-reduce operation. Section 5 provides experimental evidence of the efficiency of the proposed approach. Finally section 4 provides conclusive remarks and future research directions.

2 The K-Means Algorithm

Given a set $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ of n data vectors in a d dimensional space $\mathbb{R}^d$ and a parameter K ($1 < K < n$) which defines the number of desired clusters, k-means determines a set of K vectors $\mathscr{M} = \{\mathbf{m}_1, \ldots, \mathbf{m}_K\}$, called centres or centroids, to minimise the average within-cluster variance. The clustering associated to the set of centroids $\mathscr{M}$ is the set of disjoint partitions $\mathscr{P} = \{\mathscr{P}_1, \ldots, \mathscr{P}_K\}$, such that $\bigcup_{k=1}^{K} \mathscr{P}_k = X$ and $\mathscr{P}_i \bigcap \mathscr{P}_j = \emptyset$ ($i \neq j$). The centroid of the cluster k is derived to approximate the 'centre of mass' of the cluster partition $\mathscr{P}_k$ and is defined as:

$$\mathbf{m}_k = \left(\frac{1}{n_k}\right) \sum_{i=1}^{n_k} \mathbf{x}_i^{(k)}, \tag{1}$$

where n_k is the number of data points in the cluster k ($n_k = |\mathscr{P}_k|$), and $\mathbf{x}_i^{(k)}$ is a data point in the cluster k ($\mathbf{x}_i^{(k)} \in \mathscr{P}_k$).

The error for each cluster is the squared sum of a norm $||\cdot||$, e.g. the Euclidean norm, between each input data point and its nearest centroid. The objective function

that k-means optimises is the overall error (squared error distortion) $E(M)$ which is given by the sum of the squared errors for each cluster:

$$E(M) = \sum_{i=1}^{n} \min_{k=1..K} ||\mathbf{x}_i - \mathbf{m}_k||^2 = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \left|\left|\mathbf{x}_i^{(k)} - \mathbf{m}_k\right|\right|^2 . \quad (2)$$

The sum of the squared errors $E(M)$ is a popular objective function as it combines a measure of homogeneity and separation of the clusters. The optimal clustering corresponds to the global minimum $E(M)$. For general values of K and d and with the Euclidean distance as metric, the problem is known to be NP-hard [9].

Given an initial condition, i.e. a set of initial centroids, the k-means algorithm [2, 3] adopts a 'hill climbing' heuristic method to determine the local minimum of the objective function. The algorithm repeats an iterative refinement step until no further improvement can be achieved. At each iteration two main steps are performed:

- **distance calculation:** for each data point compute the distance to each cluster centroid and find the closest cluster centroid;
- **centroid update:** recompute each cluster centroid as the average of data points assigned to the cluster partition.

The straightforward implementation of the algorithm is often referred to as a 'brute force' approach, because it performs a 'nearest neighbour' search among the K centres for each of the n data points of the dataset. Thus, it performs exactly $n \cdot K$ distance computations at each iteration.

3 Parallel K-Means

Parallel clustering algorithms have been extensively studied (e.g. [11], [10], [6]). In particular, [6] proposes a straightforward implementation of the brute force k-means algorithm for distributed memory systems. The input data points are partitioned in equal sized sets and distributed to p processes. Initial centroids are generated at a master process and distributed (broadcast) to the other processes. Each process performs a k-means iteration on its local data partition. At the end of each iteration a global reduction operation (e.g. *MPI_ALLREDUCE* generates the updated global centroid vectors and the global distortion measure. At each process and for each iteration three main steps are performed:

- **distance calculation:** for each data point of the local data set compute the distance to each cluster centroid and find the closest cluster centroid;
- **global reduction operation:** all nodes perform a deterministic reduction operation to compute cluster global sums and counts and the global error;
- **centroid update:** recompute each cluster centroid as the average of all data points assigned to the cluster using the global sums and counts.

In a static and homogeneous computing environment of limited size the brute force approach in [6] guarantees a perfectly balanced load among the processes. At each iteration a single all-reduce operation can be used to compute the aggregation for

$(d \cdot K + 1)$ real values and K integer values over all processes. The number of communication steps of the all-to-all reduction operation is $log_2(p)$. However, this global deterministic reduction operation is not suitable for very large systems.

3.1 Parallel KD-Tree K-Means

A parallel formulation of the k-means algorithms based on KD-trees [7] can reduce the number of distance computations and the total computation cost. However, it introduces load imbalance in contrast to the perfectly balanced approach of the parallel 'brute force' approach [6]. The work in [8] provides an efficient and scalable parallel formulation based on KD-trees which employs a dynamic load balancing policy and provides better scaling properties than the parallel 'brute force' approach for large-scale and heterogeneous computing environments.

The parallel k-means algorithm based on KD-trees follows a computation - communication pattern similar to [6]. The main difference is that a distributed KD-tree is constructed (sequentially [12] or in parallel [13]) in a preprocessing step and is adopted in the core computation at each iteration to reduce the overall number of distance computations. The effectiveness of the sequential KD-tree k-means approach depends on the similarity (spatial aggregation) of the input patterns. The relative effectiveness of the parallel algorithm depends on how the data set is partitioned among the processes. The spatial aggregation of patterns must be preserved in the local partitions. To this aim in [8] appropriate data partitions are generated by exploiting the KD-tree itself. An initial KD-tree is built up to the level $log_2(p)$, where p is the number of processes. This generates p leaves with data partitions which have a spatial aggregation as good as in the sequential algorithm. The leaves of this initial KD-tree define the data partitions to be distributed to the parallel processes. Each process builds a local KD-tree from the assigned leaf of the initial tree.

The parallel KD-tree k-means approach [8] adopts this partitioning method to reduce the overall computation cost. However, it is still based on a global deterministic reduction operation, which hinders the adoption of any of these parallel k-means algorithms in large-scale distributed environments.

This work identifies in the parallel KD-tree k-means approach the opportunity to introduce an optimisation in terms of the communication cost as well.

4 Efficient Reduction Operation

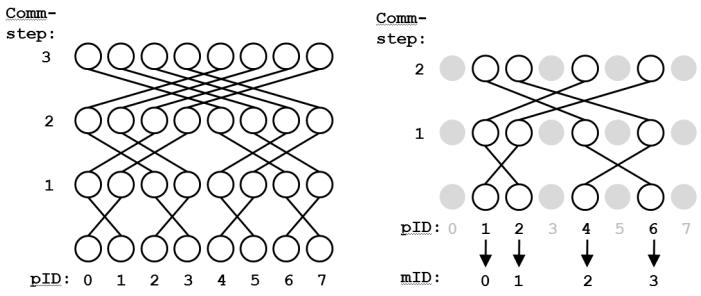
The partitioning based on KD-tree induces a non-uniform data distribution and processes do not equally contribute to the determination of the centres updates as it was in [6]. Considering a single centroid, only a subset of processes contributes patterns to its cluster. Moreover, this limited scope increases during the iterations, as the centroids better approximate the centres of mass of the clusters.

The single global all-reduce operation for all clusters can be replaced by multiple cluster-specific operations (*group-reduce*). These operations can be executed

in parallel and independently from each other. And, above all, each *group-reduce* operation can be performed over a subset of the processes.

A global all-reduce operation requires exactly $log_2(p)$ communication steps. The maximum number of the communication steps for completing any *group-reduce* operation is $log_2(p')$, where $p' = max_k(|P_k|)$ and P_k is the subset of processes which contribute to cluster k. For large-scale systems and non-uniform cluster distributions we expect that $|P_k| << p$, for each k and $p' << p$.

The global all-reduce and the *group-reduce* operations are depicted in figure 1. In the *group-reduce* operation the list of process identifiers (pID) is used to map process IDs to group IDs (mID). The *group-reduce* then is equivalent to an all-reduce operation within a smaller communication world. The *group-reduce* could be implemented by means of an all-reduce operation within an MPI communication world defined over the group. However, the definition of a new communication world would add communication overhead. Since groups may change at every k-means iteration, this approach is unlikely to provide the expected improvement.

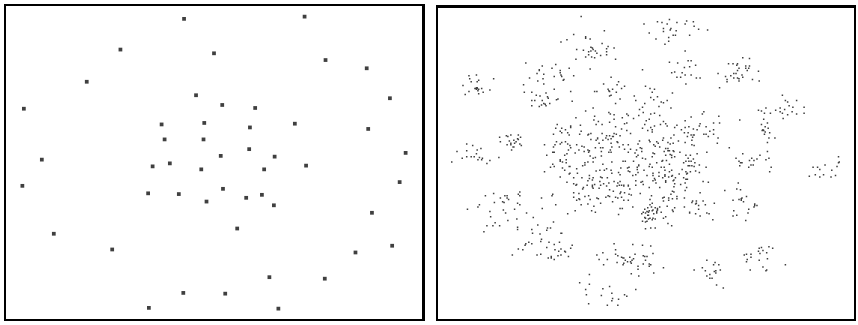


(a) all-reduce (b) group-reduce

Fig. 1 Recursive doubling for all-reduce and group-reduce

The proposed approach is based on a dynamic group membership management which does not involve additional communication. The control information for the dynamic management of group membership is piggy-backed in the *group-reduce* messages. Any change to the list is performed within the reduction operation itself; no additional communication is required. During a reduction operation the pID list is propagated and any change is incorporated. At the end of the *group-reduce* operation all processes receive the same list, which is used to define the process group for the subsequent reduction operation.

Here we briefly describe the group membership protocol. For each cluster a group leader is identified and explicitly manages initial group memberships. Each process p_i keeps a reference list of current group members L_i^{cur}, i.e. the next *group-reduce* operation is performed over the processes in it.



(a) prototypes (b) patterns

Fig. 2 2D representation of the multi-dimensional prototypes (a) and patterns (b)

The reference list (L_i^{cur}) is initialised and propagated by the group leader. Alternatively, the initial list can include all processes by default. In this case a first global all-reduce would be performed.

At each communication step s of a *group-reduce* operation the process p_i generates a local pID list $L_{i,s}$ ($L_{i,0} = L_i^{curr}$) to be included in the *group-reduce* data message. At step s a message exchange is performed with process j. The computation of the reduction operator includes the merging of the local and remote lists, $L_{i,s+1} = L_{i,s} \oplus L_{j,s}$.

When the contribution of a process is below the threshold T_{min}, it will still participate to the current group-operation and will remove its pID from the local list $L_{i,s}$: it will not participate to the next operation.

An 'external' process is a process that is not participating to the group operation, i.e. its $pID \notin L_i^{cur}$. An external process can join the group by sending its contribution asynchronously to any of the participating nodes, a contact process (e.g. the group leader). The contact process includes the external data contribution to its own for the next reduction operation and adds the external process ID to its local list $L_{i,s}$. All participating processes will receive the external process ID, which can join the group in the next *group-reduce* operation. The contact point acknowledges the join request by sending the reduced IDs list of the group to the external process.

5 Experimental Analysis

The better scalability of the parallel KD-tree k-means depends on the efficiency of the *group-reduce* operation with respect to the all-reduce operation adopted in the straighforward parallel formulation. The improvement in terms of the communication costs and running times can only be appreciated for very large and extreme-scale systems. For relatively small parallel computing systems the approach based on a *group-reduce* operation is not expected to produce a significant improvement

because of the logarithmic relation between number of processes and communication steps. Nevertheless, we can evaluate the effectiveness of the approach even for a small-scale system by determining the percentage of processes involved in the *group-reduce* operations as the all-reduce operations always involve all processes.

5.1 Data Generation and Execution Environment

We have generated an artificial data set with 200000 patterns in a 20-dimensional space with a mixed Gaussian distributions. First we have generated $K = 50$ prototypes in the multi-dimensional space. For each prototype we have generated 4000 patterns using a multivariate Gaussian distribution with a random standard deviation in the range $[0.0, 0.1]$. In order to create a more realistically skewed data distribution, we have distributed 25 prototypes uniformly in the whole domain and 25 prototypes were restricted to a subdomain. This generated a higher density of prototypes in the subdomain. The parameters were chosen in order to generate a dataset which contains some well separated clusters and some not well separated clusters in the subdomain. We have applied Multi-Dimensional Scaling to the set of prototypes and to a sample of the patterns to visualize them in 2-dimensional maps. Figures 2a and 2b show 2D maps, respectively, of the 50 prototypes and of a sample of the patterns. The higher density area is clearly visible at the center of the map.

The 20-dimensional patterns have been organised in a KD-tree and have been distributed among the processes using the nodes at level $log_2(p)$ of the initial tree. Each local partition contains an equivalent number (3125) of patterns. This can be easily achieved by adopting an opportune policy in the construction of the tree (e.g. the median approach in KD-tree [12]).

The software has been developed in Java and adopts MPJ Express [14], a Java binding for the MPI standard. The experimental tests were carried out in a IBM Bladecenter cluster (2.5GHz dual-core PowerPC 970MP) connected via a Myrinet network, running Linux (2.6.16.60-0.42.5-ppc64) and J2RE 1.5.0 (IBM J9 2.3).

5.2 Results

We have extracted and analysed some statistics on the contribution of the processes ($p = 64$) to the *group-reduce* operations for the computation of the cluster centres ($K = 50$) during the iterations of the parallel KD-tree k-means algorithm.

Figure 3 shows the number of processes actively involved in the K *group-reduce* operations. The chart shows the empirical cumulative distribution function (*cdf*) for a few first iterations $(1, 2, 3, 5)$ and the last iteration (68) of the k-means algorithm. At the very first iteration the number of processes actively involved in most group operations is 48 and is indicated by the rightmost vertical segment on the chart. The initial centroids that are used during the first iteration are chosen randomly and the contribution of many processes is required, although they are already less than the total number of processes (75%).

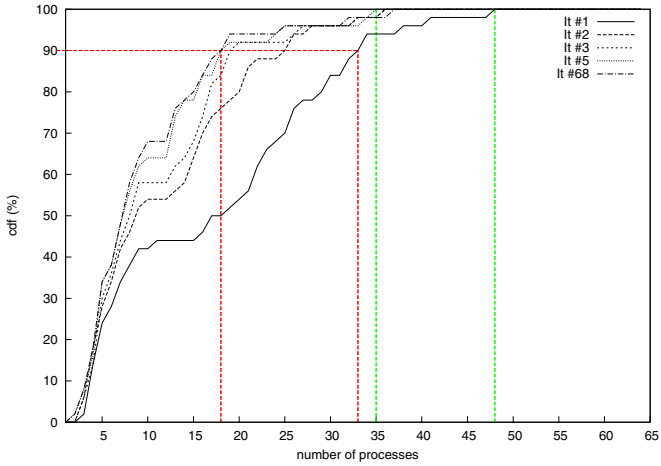


Fig. 3 Processes in the *group-reduce* operations: exact reduction at different iteration cycles

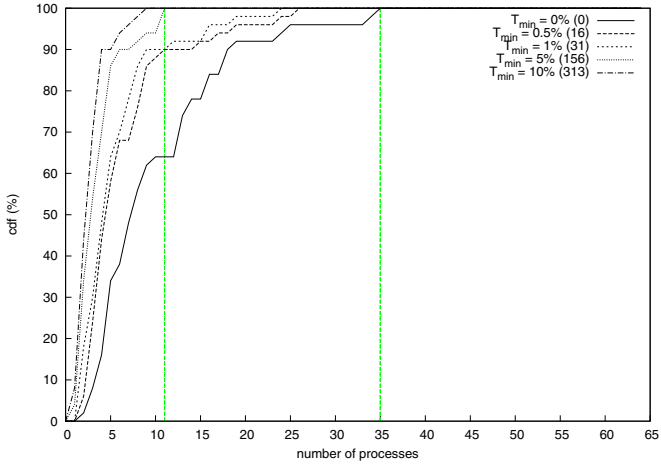


Fig. 4 Processes in the *group-reduce* operations at It#5: exact and approximate redutions

It is interesting to notice that after the first iteration less processes are actively involved and this is quickly converging towards the behaviour at the last iteration. The curves for the iterations 5 and 68 are barely distinguishable. From the the second iteration the centroids already are an approximation of the centre of mass of the clusters. Given the good spatial aggregation in each local partition provided by the KD-tree partitioning method, many local partitions will contribute only to a subset of the clusters: some clusters will receive small contributions or any contribution at all by many processes.

In the exact parallel k-means algorithm all contributions have to be taken into account regardless of how small they are. For this reason a few group-operations

still require a high number of processes (48 at It#1, 35 at iteration 35), although some contributions contain very few patterns. In iteration #1 and #5 a few processes, respectively, 33 and 18, account for 90% of the contributions.

A straightforward optimisation is then the introduction of an approximation error to inhibit processes from giving small contribution to group operations. We have adopted a threshold T_{min}, defined as the minimum percentage of the local data patterns which a process can contribute to a group operation.

Figure 4 shows the empirical *cdf* for various T_{min} values. If we consider $T_{min} = 5\%$, only those processes contributing with at least 156 data patterns (over 3125) to a cluster participate to the corresponding group-reduction operation. In this case, any operation involves no more than 11 (p') processes instead of 35 of the the exact algorithm and 64 (p) of the global all-reduce approach. The maximum number of communication steps for any *group-reduce* operation is $\lceil log_2(p') \rceil = 4$ instead of $log_2(p) = 6$, leading to an improvement of 33.3%.

6 Conclusions

Many parallel data mining algorithms can not be currently adopted in very large-scale systems for the poor scalability property of the adopted communication patterns, such as global reduction operations. We have presented a communication-efficient parallel formulation of the k-means algorithm for cluster analysis. The approach is based on multi-dimensional binary search trees (KD-trees) and a novel *group-reduce* operation. In contrast to the global all-reduce operation, *group-reduce* dynamically adapts and restricts the communication group to those processes which provide useful contributions to the operation. KD-trees are used to induce a non-uniform data distribution among the processes to maximise the advantage of the *group-reduce* operation. The experimental analysis in a small-scale environment with 64 processors has confirmed that the method is effective in reducing the number of communication step for each k-means iteration. Although, the advantage is expected to be significant only for very large-scale systems due to the log relation between the number of processes and the number of communication steps of the reduction operation. Further work will be devoted to the simulation of very large-scale scenarios to assess the reduction of the communication costs.

References

1. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recogn. Lett. 31(8), 651–666 (2010)
2. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: Proc. of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
3. Lloyd, S.P.: Least squares quantization in PCM. IEEE Trans. on Information Theory IT-28(2), 129–137 (1982)

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 PAMI 24(7), 881–892 (2002)
5. Pettinger, D., Di Fatta, G.: Space partitioning for scalable k-means. In: Proc. of the 9th Int.l Conf. on Machine Learning and App. (ICMLA), pp. 319–324 (2010)
6. Dhillon, I.S., Modha, D.S.: A Data-Clustering Algorithm on Distributed Memory Multiprocessors. In: Zaki, M.J., Ho, C.-T. (eds.) KDD 1999. LNCS (LNAI), vol. 1759, pp. 245–260. Springer, Heidelberg (2000)
7. Pettinger, D., Fatta, G.D.: Scalability of efficient parallel K-Means. In: Proc. of the 5th IEEE Int.l Conf. on e-Science, Workshop on Computational e-Science, pp. 96–101 (2009)
8. Fatta, G.D., Pettinger, D.: Dynamic load balancing in parallel KD-Tree K-Means. In: Proc. of the Int.l IEEE Conf. on Scalable Computing and Communications (ScalCom), pp. 2478–2485 (2010)
9. Aloise, D., Deshpande, A., Hansen, P., Popat, P.: NP-hardness of euclidean sum-of-squares clustering. Machine Learning 75, 245–248 (2009)
10. Judd, D., McKinley, P.K., Jain, A.K.: Large-scale parallel data clustering. IEEE Transactions on PAMI 20(8), 871–876 (1998)
11. Foti, D., Lipari, D., Pizzuti, C., Talia, D.: Scalable Parallel Clustering for Data Mining on Multicomputers. In: Rolim, J.D.P. (ed.) IPDPS-WS 2000. LNCS, vol. 1800, pp. 390–398. Springer, Heidelberg (2000)
12. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9), 509–517 (1975)
13. Al-furaih, I., Aluru, S., Goil, S., Ranka, S.: Parallel construction of multidimensional binary search trees. IEEE Trans. on Parallel and Distributed Systems 11(2), 136–148 (2000)
14. Baker, M., Carpenter, B., Shafi, A.: MPJ Express: Towards thread safe Java HPC. In: Proc. of the IEEE Int.l Conf. on Cluster Computing (2006)

A High Performance Engine for Concurrent CEP in Erlang

Bill Karakostas

Abstract. This paper describes the architecture, prototype implementation and performance analysis of a complex event processing engine that can meet soft real time constraints, and scale up to many thousands of concurrent events. We have avoided the resource overheads associated with the processing of large numbers of concurrent events (such as, for example, those generated by user sessions in a web site) by using lightweight Erlang processes and a 'shared nothing' architecture. We demonstrate how this approach can achieve predictable event processing times under varying loads, using only modest hardware resources in an SMP architecture.

1 Introduction

The need to detect and respond to events forming complex patterns (CEP) under (soft) real time constraints is important in systems and applications, such as process control and telecommunications. In such systems, soft real time constraints must be met in order for the system to cope with large numbers of simultaneous events (for example, phone calls) while maintaining satisfactory response times.

The ability to meet soft real time constraints imposes specific requirements for the architecture and design of a complex event processing engine (CEP). The allocation of computing resources to the processing of individual events must take into account the number of concurrently occurring events and the processing required for each event. If shared resources such as databases, are involved, care must be exercised to avoid situations such as race conditions that will hamper system performance. Performance constraint satisfaction is usually very hard, or even impossible, to verify by static means, i.e. without stressing the system under realistic load conditions.

In this paper we propose an approach in which each occurring event pattern (i.e. a cluster of related events) is assigned to a dedicated computing resource

Bill Karakostas
City University, London, UK
e-mail: billk@soi.city.ac.uk

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 165–170.
springerlink.com

(a lightweight process) that operates in a 'share nothing architecture'. By ensuring that processing of such event patterns does not require access to shared resources such as I/O and databases (i.e. it is 'side effect free'), performance can be more predictable for a given execution environment.

Moreover, the event patterns considered in this paper comprise (relatively) few events that are temporally close to each other; thus a process that handles an event pattern executes for a brief time and does not hog system resources.

The remaining of the paper is structured as follows. Section 2 surveys related work in the area of complex event processing languages, architectures and performance estimation approaches. The next section describes the design rationale architecture of the CEP engine, while its main components such as the event description language and the syntax of Event Condition Action rules are described in Section 4. Section 4 also presents performance statistics in a number of experiments, in a simulated multiuser environment. Finally, Section 5 concludes with the main benefits of the approach and with suggestions for research for its future enhancement.

2 State of the Art

Complex event pattern languages allow the specification of complex events in the condition part of Event-Condition-Action (ECA) rules. To describe relationships between times of different events, either absolute or relative, suitable expressive languages are required, typically based on some kind of temporal logic. Such event languages have been proposed for example in [1] and [3]. Performance of CEP engines has been critical for real-time computing and analytics applications . Equally important is the ability to scale performance in line with system load in order to deliver continued performance over time. There are however relatively few approaches reported in the literature, where performance constraints are explicitly accommodated by the architecture of a CEP engine. One such approach reported in [2] is essentially a framework that generates code for a CEP application and enables to determine time bounds on the application response to a set of supported events. The approach presented here, achieves the required response times, using modest hardware resources, and can scale up to large numbers of concurrent users, i.e. in the region of 10000 using the experimental prototype described in this paper.

3 Architecture of a CEP Engine in Erlang

The high level architecture is shown in Figure 1. The core engine is implemented as an Erlang module that implements functions for receiving events and dispatching them to spawned processes. Another module contains the ECA rules, with each rule also implemented as an Erlang function. At runtime, the core engine uses two dynamic, memory based, structures (*dictionaries* in Erlang terminology) that contain the hashes of event types to rules, and the currently active rule instances, hashed by user session keys. All modules of the above architecture operate within a single Erlang VM ('Emulator').

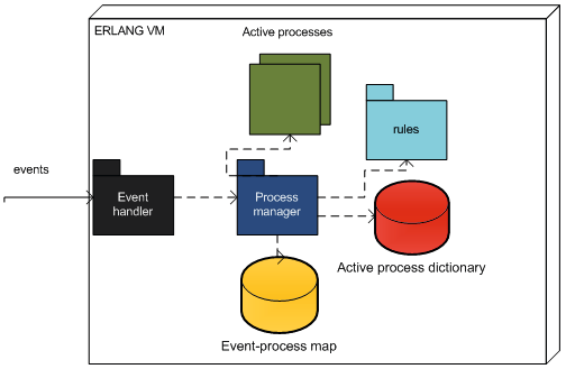


Fig. 1 Architecture of the CEP Engine

Both events and ECA rules are implemented using native Erlang data types and computational structures. Although several ECA rule notations exist, for example Rule-ML [5], none of them was adopted in order to avoid the overheads incurred by the need for a rule interpreter/parser and associated data and format conversions.

An ECA rule is an Erlang function consisting of a number of clauses that update the current condition part (i.e. state) of the rule based on the type of the received events. When the current rule condition matches the expected rule condition, the rule is fired. Each ECA rule essentially implements a business rule (an advertising or marketing strategy for example) of the company that utilises the event engine.

An example of how ECA rules are defined in Erlang is shown below. The rule below will fire if three events of types e1, e2, e3 are received (in no specific order) within 10000 milliseconds from the time the rule is spawned.

```
ecarule1234(E1, E2, E3) ->
case (E1 == e1) and (E2 == e2) and (E3 == e3) of
true -> ruleaction1234(e1,e2,e3), exit(0);
false ->
receive Event
case
e1 -> ecarule1234(e1, E2, E3);
e2 -> ecarule1234(E1,e2,E3);
e3 -> ecarule1234(E1,E2,e3);
Other -> ecarule1234(E1,E2,E3)
after 10000 -> exit(0)
end.
```

4 Performance Evaluation

As argued in Section 2, meeting strict response constraints in an ECP driven system can be challenging and is, in general, hard to verify. The total reaction time to an event in a CEP system can be broken down into the following constituents:

1. td: the time it takes for a single event to be detected and validated by the system
2. te: the time it takes for the ECA rule engine to evaluate the pattern and fire an action if there is a match
3. ta: the time it takes to execute the action and update the relevant part of the ECP driven system.

The performance constraint can be formulated as td + te + ta $< t_{max}$ where t_{max} is set according to the type of the target ECP driven system.

Assuming that the CEP system has the following average performance characteristics:

- Event detection rate d (events per CPU cycle) or C * d time units, (assuming a CPU clock of C cycles per time unit)
- ECA rule evaluation rate r (rules per CPU cycle) or C * r time units
- Action rate a (actions per CPU cycle) or C * a time units

A sequential processing of a load of n event patterns would mean that the n_{th} event pattern will be processed in n*(1/d + 1/r + 1/a) cycles or n*(1/d + 1/r + 1/a)/C time units (e.g. seconds).

To meet the timing requirements for all events we must ensure that n*(1/d + 1/r + 1/a)/C $< t_{max}$

One obvious way to meet such performance targets is to increase CPU performance (instructions per clock cycle) and/or clock speed. The alternative as advocated in this paper is to execute concurrently one or more of the processing steps, d, r, a. Thus, in this paper the rule evaluation and firing is proposed to be executed concurrently, ensuring that (within the overall resource limits) the handling of each event cluster/user session is independent from that of the other event clusters.

To understand real performance of the above architecture a prototype event engine was implemented in Erlang (version R15b). The prototype implementation consisted of the CEP engine, 10 sample rules and an Erlang application that simulates users interacting with the system.

We executed the prototype on a Windows 7 PC with an Intel Core 2 Duo T7300 2Ghz, dual core processor and 2 gigabytes of RAM. A version of the Erlang emulator with symmetric multiprocessing capabilities, was deployed, in order to utilise the multicore architecture of the host platform.

Ten distinct event types and ten sample rules with no side effects (I/O and database access) were used in the experiments. A separate Erlang application that simulates connected users by randomly generating events, was used to stress the system and to measured average event pattern handling times. Figure 2 shows that CPU utilisation by the different CEP engine components grows linearly with the number of connected users. For 1000, 5000 and 10000 concurrent user sessions, average response time was almost constant and in the region of 150 seconds. To achieve lower response rates, CPU speed or the degree of concurrency in rule execution, would need to be increased, as explained in Section 3.

Performance of CEP engines is improving steadily, mainly due to advances in hardware. As an example, the Real-Time Event Engine (RTEE) from Object Design

[4] reported the ability to capture over 11,000 network events/second on a single processor mid-range workstation, while simultaneously querying real-time events to make them available to operational support systems. More recent CEP engines that utilise dedicated hardware architectures (both parallel and distributed) have improved even further on these figures. However, it is not always obvious how such systems can scale up and exploit advances in hardware and software architectures without radical redesign of their architectures. The prototype architecture presented in this paper compares therefore favourably with current commercial developments, but more importantly addresses the scalability potential by exploring the concurrency potential that is presented in the new generation of multicore computer architectures and the inherent capability to exploit them, that is built in development environments such as Erlang/OTP.

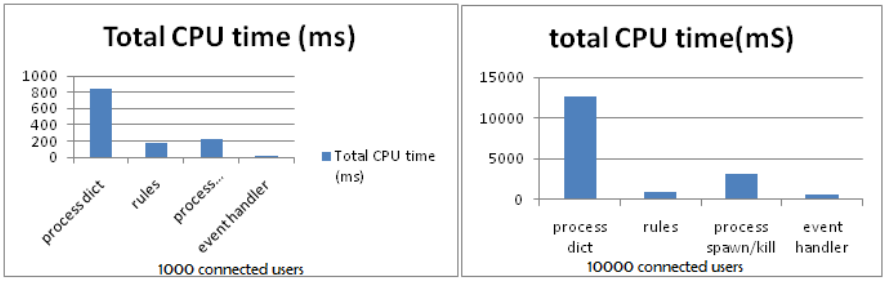


Fig. 2 CPU utilisation by the CEP Engine

5 Conclusions

This paper has argued that event handling in soft real time is possible with commodity systems of modest resources, even when the numbers of concurrent events are very large (i.e. in the order of tens of thousands per second), by utilising concurrent lightweight computations.

In the event engine architecture described in this paper, we avoid the performance penalty associated with the creation of multiple threads to handle user sessions, by utilising the Erlang language's native threads that execute within a virtual machine and consume substantially fewer system resources compared to operating system threads.

The category of systems we envisage been supported by this architecture, are Web sites (i.e. for ecommerce, social media etc.) where users attention must be captured (online marketing) and purchases can be influenced. Although individual user's behaviour cannot be predicted, the average time users spend on a site can be statistically calculated. Thus, it is possible to estimate the time a user will remained connected, and therefore the required response time. As the average time a user stays connected to a web site can be measured in only a few tens of seconds for some types of web

sites, the ability to be able to handle thousands of connected user sessions with under 10 second response times is often required from today's CEP engines.

An additional advantage of systems like the one described here is that both CEP engine and rule development share the same programming environment (Erlang), thus avoiding impedance mismatch. For even more seamless integration of the CEP and the monitored system, a browser could operate within the same (Erlang) environment as the engine, such as for example the Yaws web server which is written in Erlang and uses Erlang as its embedded language. Additional mechanisms for storage of user sessions both for long term purposes (i.e. mining) and for recovery from temporary failures at runtime need also to be explored.

References

1. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002)
2. Magid, Y., Oren, D., Botzer, D., Adi, A., Shulman, B., Rabinovich, E., Barnea, M.: Generating real-time complex event-processing applications. IBM Systems Journal Archive 47(2) (April 2008)
3. Mansouri-Samani, M., Sloman, M.: GEM: A generalised event monitoring language for distributed systems. Distributed Systems Engineering 4(2) (1997)
4. Object Design. Object Design Introduces Real-Time Market Data Application for the Financial Industry; Captures 10,000, and Queries More Than One Million, Market Data Events Per Second (2001), http://www.thefreelibrary.com/Object+Design+Introduces+Real-Time+Market+Data+Application+for+the...-a074517138
5. Paschke, A.: ECA-RuleML: An Approach combining ECA Rules with temporal interval-based KR Event/Action Logics and Transactional Update Logics, IBIS, Technische Universität Mün-chen, Technical Report 11-2005

Contextual Synchronization for Efficient Social Collaborations: A Case Study on TweetPulse

Jason J. Jung

Abstract. It is important to be aware of user contexts for supporting efficient collaborations among them. The goal of this paper is to present a social collaboration platform where we can understand *i*) how the user contexts are dynamically changing over time, and *ii*) how the user contexts are mixed with multiple sub-contexts together. Thereby, we have implemented TweetPulse, which is a a Twitter-based tool for context monitoring and propagation system in a given social network. TweetPulse can match contexts of the users (and integrate them) to find the most relevant users. Eventually, collaboration among users are *contextually synchronized*. by dynamically organizing a number of communities. A set of users in each community come together to share skills or core competencies and resources at the moment. We have shown the experimental results collected from a collaborative information searching system in terms of *i*) setting thresholds, *ii*) searching performance, and *iii*) scalability testing.

1 Introduction

Computer-supported collaborative work (CSCW) has been important to deal with complex problems in many different domains. Particularly, in virtual organizations (e.g., virtual enterprises and e-learning systems), a large number of communications are needed for supporting collaborations among multiple users. Main strategy of the existing CSCW schemes and systems is to discover user communities (called collaborative networks), with respect to the contexts of the users. Thus, users can more efficiently collaborate with other users in the same community by providing various processes and services (e.g., expert finding and question-answering).

Jason J. Jung
Department of Computer Engineering
Yeungnam University
Gyeongsan, Korea
e-mail: j2jung@gmail.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 171–179.
springerlink.com

However, they assume that the user contexts should be fixed and static. It means that the collaboration is not possible any more, after the user has been involved to different task. The contexts of users are dynamically changing over time. The CSCW systems have to be able to take into account the dynamics of contexts [2, 3, 7, 9, 10].

In previous work, we have claimed that *contextual synchronization* is useful for ontology-based collaboration [7]. Additional empirical studies on contextual synchronization shows that the proposed work outperforms in terms of scalability [8].

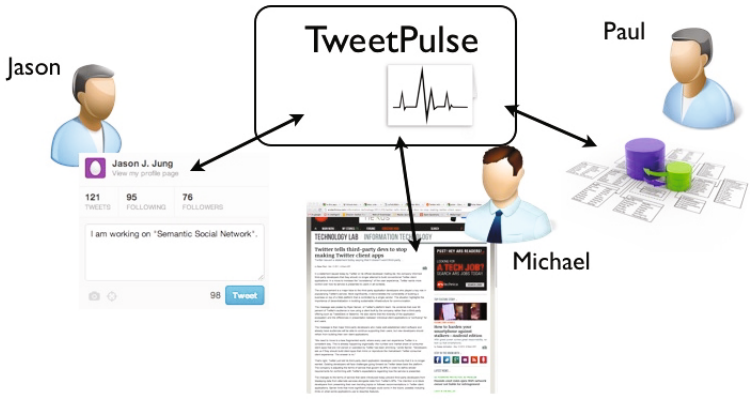


Fig. 1 TweetPulse for Contextual Synchronization

Various social media tools (e.g., Twitter and FaceBook) have been used to communicate with other users. Especially, many literatures have found that such social media tools are powerful for propagating information to multiple users. Thus, the goal of this work is to exploit the social media tools for supporting contextual synchronization process. As shown in Fig. 1, given a set of three users, the latest contexts of other collaborators can be collected through not only directly the corresponding social media (e.g., Twitter) but also additional media channels (e.g., webpages and third-party applications). Then, the proposed system (called TweetPulse) can find the contextual relationships between the users, and eventually, organize a set of communities for supporting their collaborations.

The outline of this paper is as follows. In Sect. 2, we defines several notations for representing user contexts. Sect. 3 gives a main contribution on contextual mediation among TweetPulse users. Sect. 4 shows how to evaluate the proposed CSCW architecture. Finally, in Sect. 5, we draw a conclusion of this work.

2 Context Modeling

The user contexts are extracted from social activities in the proposed system. In this section, we want to discuss how to formulate the ontologies, user contexts and

mappings between the contexts by using the notations introduced in our previous work [7, 8].

Basically, personal ontologies are applied to represent their own domains and interests. More importantly, these formal representations make the user contexts to be comparable.

Definition 1 (Ontology). An ontology $\mathbb{O}$ is represented as

$$\mathbb{O} := (\mathscr{C}, \mathscr{R}, \mathscr{E}_{\mathscr{R}}, \mathscr{I}_{\mathscr{C}}) \tag{1}$$

where $\mathscr{C}$ and $\mathscr{R}$ are a set of classes (or concepts) and a set of semantic relations (e.g., equivalence, subsumption, disjunction, etc), respectively. $\mathscr{E}_{\mathscr{R}} \subseteq \mathscr{C} \times \mathscr{C}$ is a set of relationships between classes, represented as a set of triples $\{\langle c_i, r, c_j \rangle | c_i, c_j \in \mathscr{C}, r \in \mathscr{R}\}$, and $\mathscr{I}_{\mathscr{C}}$ is a power set of instance sets of a class $c_i \in \mathscr{C}$.

Each user can take actions (e.g., annotation by creating his own personal ontology) during managing his resources (e.g., blogs, bookmarks, and KMS) in personal information repositories.

Definition 2 (Personal ontology). For a given set of annotations $\mathscr{A}_k$ from resources RES_k of user u_k, his personal ontology $\mathbb{O}_i^{\mathscr{P}}$ is represented as

$$\mathbb{O}_k^{\mathscr{P}} = (\mathscr{C}, \mathscr{R}, \mathscr{E}_{\mathscr{R}}, \mathscr{I}_{\mathscr{C}}, \mathscr{A}_k) \tag{2}$$

where $\mathscr{A}_k = \{\langle c_i, r, res_j \rangle, \langle ins_\alpha, r, res_j \rangle | c_i \in \mathscr{C}, ins_\alpha \in \mathscr{I}_{\mathscr{C}}, r \in \mathscr{R}, res_j \in RES_k\}$.

Current context depends on the resources the user is working on for the moment. Also, the semantics are determined by the mapping with his personal ontology.

Definition 3 (Context). At a certain moment t, a context $ctx_k^{(t)}$ of user u_k obtained by a mapping function $\mathscr{M}$ at time t is given by

$$ctx_k^{(t)} = \{c_i | c_i \in \mathscr{M}(\mathbb{O}_k^{\mathscr{P}}, res^{(t)})\} \tag{3}$$

where $res^{(t)}$ is a resource that the user is working on.

Definition 4 (Mapping). A mapping function $\mathscr{M}$ is given by

$$\mathscr{M}(\mathbb{O}_k^{\mathscr{P}}, res^{(t)}) = \arg_{\{c_i | \langle c_i, r, a \rangle \in \mathbb{O}_k^{\mathscr{P}}\}} \max \frac{\sum_{f \in \mathscr{S}(res^{(t)}), a \in \mathscr{A}} Sim_{\mathscr{L}}(f, a)}{|\mathscr{S}(res^{(t)})|} \tag{4}$$

where $Sim_{\mathscr{L}} = 1 - \frac{Distance(f,a)}{\max(|f|,|a|)}$ measuring the similarity between two given terms f and a based on string matching algorithms. This function returns a set of concepts c_i applied to annotation a. The sting matching can be based on various schemes, e.g., edit and levenshtein distances. For simplicity, substring distance has been used in this work.

Thus, by comparing these user contexts, a set of collaborative networks can be built. We can assume that all of the user contexts are contextually cohesive.

Definition 5 (Collaborative network). A collaborative network CN_i consists of

$$CN_i = \langle \mathscr{U}_i, \mathscr{V}_i \rangle \tag{5}$$

where $\mathscr{U}_i$ is a set of users and $\mathscr{V}_i \subseteq |\mathscr{U}_i| \times |\mathscr{U}_i|$.

Definition 6 (Group context). Given a collaborative network CN_i, a group context is represented as

$$ctx^{\top}_{CN_i} = \bigcap_{k=1}^{|\mathscr{U}_i|} ctx_{u_k} \tag{6}$$

where $u_k \in CN_i$. It means a set of the most concepts covering all of the personal user contexts.

Additionally, in a collaborative network, social centrality of users can be computed. Then, the most centralized user (denoted as $u^{\diamond}_{CN_i}$) is regarded as capable of playing the role of a contextual representative of the corresponding CN. Compared to incremental building of consensus ontologies over time [6], this group context must be updated whenever people have a new context.

3 Contextual Collaboration

In this work, the contextual collaboration should be synchronized. It is regarded as a community identification process, with respect to the latest user contexts. Once we have a set of user contexts at a certain moment, the user contexts and the relationships between the contexts should be considered to support efficient collaborations in real time.

3.1 Contextual Synchronization for Ad Hoc Collaboration

Since particularly the user context is dynamically changing, the community identification process should be done in an ad hoc manner. Thus, the contextual synchronization procedure [7] consists of three basic steps; *i*) integrating personal contexts into a group context, *ii*) detecting contextual transitions, and *iii*) re-organizing CNs for a new group context.

Particularly, to be aware of contextual transition of the users, we are considering that the context of a user can be changed while conducting a certain task. Thus, whenever any contextual transition of the context is detected, the CN must be re-organized. Context transition is based on comparison between a context $ctx^{(t)}$ and previous ones, e.g., $ctx^{(t-1)}$. If the difference is above a threshold τ_{CTX}, we assume that the corresponding user (or users) is researching distinct resources. For instance, the function for user u_k may be formulated by testing the following step;

$$Sim_{CTX}(ctx^{(t)}_{u_k}, ctx^{(t-1)}_{u_k}) \leq \tau_{CTX}. \tag{7}$$

Instead of the contexts of every individual user, the group contexts of CNs are applied to enable greater efficiency in the testing calculation.

More importantly, three semantic factors (i.e., semantic distance matrix $\Delta^{\diamond}$, semantic distance mean $\mu^{\diamond}$, and semantic distance deviation $\sigma^{\diamond}$) are defined to measure temporal patterns indicating relationships between contexts of users in a CN, and group contexts [7]. We mainly focus on a sequence of contexts $[ctx^{(t-T+1)}, ctx^{(t)}]$ by using the sliding windows method, where T is the size of the time interval.

Statistical distribution of contextual transitions can be established by repeating semantic factor computation in a given time interval, over time. We have to point out semantically significant transition moments of the contexts during a certain task based on temporal dynamics of semantic factors. Hence, particular special triggering patterns from these signals are regarded as important evidence for contextual synchronization.

For practical computation, activities of users in CN_i are aggregated during a given time interval T, and are represented as a set of concepts from personal ontologies in the form of a matrix $\mathscr{W}(CN_i)$. From this, we obtain a sequence of concept sets that are aggregated by both contextual dynamics of; *i*) a particular user's activities (ctx_{u_k} is indicated by k-th row components in $\mathscr{W}(CN_i)$) and *ii*) a group context $ctx^{\top}_{CN_i}$ obtained from merging the column components at each moment.

The contextual synchronization process is organized as two steps;

1. **Alerting step.** This is to find out a certain moment of change t_p when a context ctx_{u_k} becomes different from the corresponding group context $ctx^{\top}_{CN_i}$. Semantic distance deviation $\sigma^{\diamond}$ of column components in $\mathscr{W}(CN_i)$ is applied to derive these significant contextual transitions of a particular user u_k in CN_i. Time points t_{Alert} for alert step can be characterized by controlling the threshold value for alerting λ_{Alert}. It means that there are users u_{Alert} whose contexts turn different from the other members in a collaborative network ($ctx^{\top}_{CN_i}$ transitions).
2. **Confirming step.** Once we have discovered a set of users $u^{t_p}_{Alert}$ whose contexts are probably changed at time t_p in the previous step, the confirming step is to make sure whether users exhibit contextual transitions or not, and to discover the specific transition moments of contexts of the confirmed users by comparing the previous context. By the continuity property of ontology-based contexts, it can be discovered by using the subsequence derived from each row component of $\mathscr{W}(CN_i)$. If users u_{Alert} have exhibited any contextual transitions, we can detect more specific time points t_S of contextual transitions. Similar to the previous "alerting" step, the confirming step for the alerted users can be characterized by controlling the threshold $\lambda_{Confirm}$ for confirming contextual transitions of context $ctx^{(t_{sq})}_{u_j}$. Hence, a set of time points t^j_S is likely the moment when the personal context of the corresponding user changes.

We will discuss further details about *i*) controlling two thresholds and *ii*) associations between contexts of people and group context of the corresponding CN in Sect. 4. Each time user contexts within a collaborative network CN_i are stored in

$\mathscr{W}(CN_i)$, this two-step procedure (i.e., alerting and confirming) for detecting contextual transitions has been implemented [7]. We employ the semantic distance deviation $\sigma^{\diamond}$ to establish the dispersion of members in CN_i, rather than the group context itself. Subsequently, if some users are detected in this step, the confirming step can establish if their transitions are validated or not, because the semantic distance mean $\mu^{\diamond}$ is useful to measure semantic cohesion within a time interval.

3.2 *TweetPulse as a Mediator*

A practical system, called TweetPulse, has been designed and implemented to support contextually synchronized collaboration among users. TweetPulse is a system to collect user contexts from Twitter, which is the most popular social media. Fig. 2 shows a system architecture of TweetPulse, which consists 7 main components.

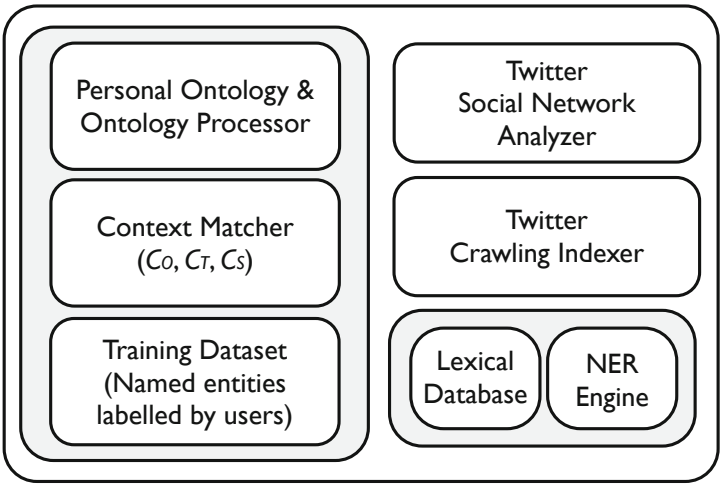


Fig. 2 System architecture of TweetPulse

Particularly, the textual data generated from Twitter is too short to precisely recognize the user contexts. Named Entity Recognition (NER) module is employed to build a set of tweets which are contextually related with each other.

- Lexical database contains all of the user-generated annotations for named entities. According to the classes, users are supported to label named entities by using the corresponding tags, i.e., ⟨PER⟩, ⟨ORG⟩, ⟨LOC⟩, and ⟨DID⟩. Also, it stores lexical resources which users can freely access and retrieve their own lexical resources by using standard queries. Once the training dataset has been established by users, the dataset is stored in this lexical database.
- NER engine is to conduct the classic NER tasks by learning lexical patterns from the previous two modules. In this work, we simply employ a maximum entropy approach [1] for the NER engine.

- Context matcher is a module for performing a set of predefined heuristics. More importantly, it has to integrate all available contextual associations computed from the heuristics, so that it can build a tweet cluster with a given latest tweet.
- Twitter social network analyzer can build an ego-centric social network, which is a directed graph structure, from followers and followings. The results from this module have to be sent to the context matcher for enriching the social association.
- Twitter crawling indexer is to extract tweets from a target user. The extracted tweets are indexed by referring to the training dataset given by the user. It is also an additional module for enriching the contextual associations.

4 Experimental Results

We have conducted several experiments to evaluate the proposed synchronization methods. Three groups (ten users in each group), G_A (simple co-browsing), G_B (Bayesian synchronization), and G_C (TweetPulse-based synchronization) were organized. They were asked to build their own personal ontologies with OWL. (Of course, it does not sound realistic. Many studies on ontology learning and user modeling have proposed to adaptively extract and formalize such user contexts.) In addition, these ontologies were enriched by annotating a given set of images[1]. Then, we collected the 30 log sequences by enabling the users to browse the testing bed with their own fixed contexts. We prepared the testing dataset after cleansing the collected dataset by the preprocessing scheme proposed in [5]. Fig. 3 shows a snapshot of how TweetPulse is collecting the contexts of two selected persons and visualize them.

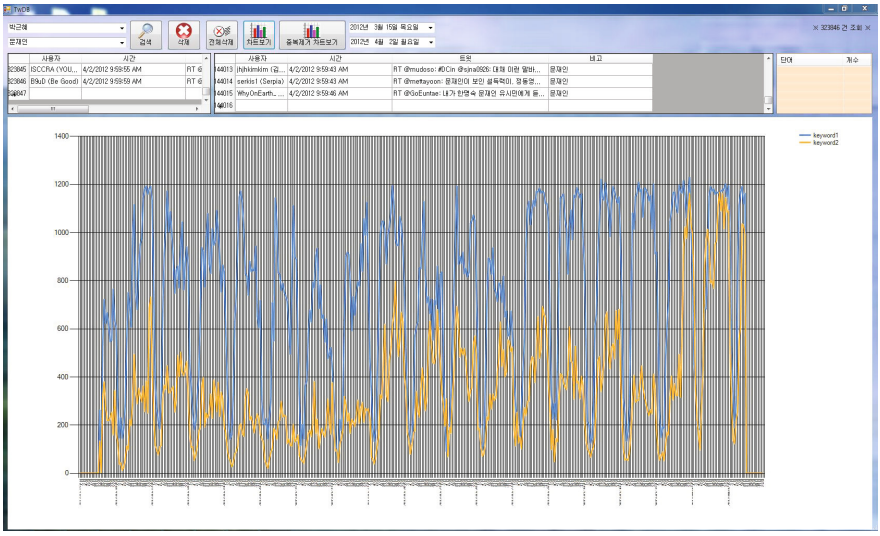


Fig. 3 A snapshot of monitoring user contexts by TweetPulse

[1] The number of images in this collection is 56,302. This collection is available in `http://intelligent.pe.kr/AnnotGrid/`

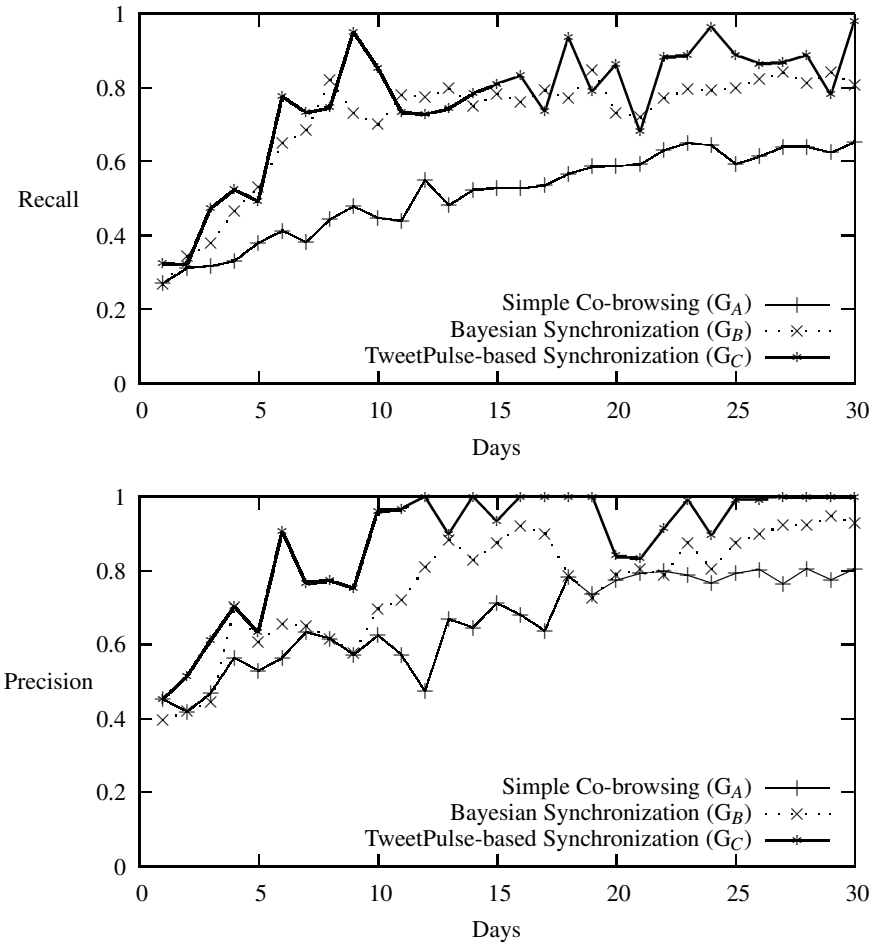


Fig. 4 Performance of information searching from three CNs with *recall* and *precision*

The evaluation issue is to show whether TweetPulse-based community identification can increase the performance of social collaboration. We have measured the efficiency of online collaboration during browsing, compared to individual browsing or basic co-browsing systems without contextual synchronization. While users in G_A browse with simple collaboration, G_B and G_C perform online collaborations, detecting context transitions dynamically. Users in G_B are supported with a recommendation scheme based on Bayesian influence propagation [4]. G_C were the only users provided with the proposed context mapping algorithm by using TweetPulse. We monitored the performance of information searching tasks in each group over time, by comparing the topics derived from the retrieved information with the topics selected prior to to experiments.

As shown in Fig. 4, G_C (contextual synchronization) exhibited the best *recall* results (in the 9^{th} day, approximately four times higher than G_A, and in the 7^{th}

day, 79% higher than G_B). This implies that our contextualized synchronization can support users, particularly during the early stages. With respect to *precision*, we discovered that collaborative systems finally showed convergence in the 80% precision level, even though, during the initial stage, individual browsing exhibited the best performance.

5 Concluding Remarks

As a conclusion, this work claims that an efficient collaboration can be fulfilled by contextual synchronization via social media (i.e., Twitter). We have implemented a practical collaborative system, called TweetPulse, and also collected an experimental result for proving that the proposed system is better than the traditional communication systems.

In future work, the system should be more elaborated, since this work is showing the on-going projects. With a large number of users, we can expect more various propagation patterns through the social media.

References

1. Berger, A.L., Pietra, S.D., Pietra, V.J.D.: A maximum entropy approach to natural language processing. Computational Linguistics 22(1), 39–71 (1996)
2. Edwards, W.K.: Putting computing in context: An infrastructure to support extensible context-enhanced collaborative applications. ACM Transactions on Computer-Human Interaction 12(4), 446–474 (2005), doi:http://doi.acm.org/10.1145/1121112.1121117
3. Gross, T., Prinz, W.: Modelling shared contexts in cooperative environments: Concept, implementation, and evaluation. Computer Supported Cooperative Work 13(3), 283–303 (2004)
4. Jung, J.J.: Collaborative web browsing based on semantic extraction of user interests with bookmarks. Journal of Universal Computer Science 11(2), 213–228 (2005)
5. Jung, J.J.: Visualizing recommendation flow on social networks. Journal of Universal Computer Science 11(11), 1780–1791 (2005)
6. Jung, J.J.: Ontological framework based on contextual mediation for collaborative information retrieval. Information Retrieval 10(1), 85–109 (2007), doi:10.1007/s10791-006-9013-5
7. Jung, J.J.: Ontology-based context synchronization for ad-hoc social collaborations. Knowledge-Based Systems 21(7), 573–580 (2008)
8. Jung, J.J.: Boosting social collaborations based on contextual synchronization: An empirical study. Expert Systems with Applications 38(5), 4809–4815 (2011)
9. Nunes, V.T., Santoro, F.M., Borges, M.R.: A context-based model for knowledge management embodied in work processes. Information Sciences 179(15), 2538–2554 (2009)
10. Schmidt, A.: Management of Dynamic and Imperfect User Context Information. In: Meersman, R., Tari, Z., Corsaro, A. (eds.) OTM-WS 2004. LNCS, vol. 3292, pp. 779–786. Springer, Heidelberg (2004)

Crafting Kinship Networks by Exploring Acquaintances Relationships

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

Abstract. In the modern knowledge society it is becoming increasingly important to maintain and promote social connections; among these, the kinship allows people to keep in touch with their relatives. In this paper we present a proposal for building and enhancing kinship networks by exploring acquaintances and friend relationships, showing the effectiveness and expandability of our approach.

1 Introduction

Maintaining and promoting social relationships is more and more considered a must in the modern knowledge society, in particular when they are transposed into the Web [4]. Such relationships are generally dynamic over time and require active maintenance [5], and many of them can be interpreted as a *complex networks* of interconnected elements [9] [6]; this allows to address issues as:

- why and how connections are established and what they mean in real world
- the measure of attitudes towards collaboration or competition among individuals or group of individuals (including communities)
- the analysis of the diffusion of information and knowledge
- the assessment of links evolution over time (network dynamics)

Several types of social relationships can be considered (e.g. customer-seller, friend-friend, and so on) [6]; a specific type of social connection is the kinship, that allows users to keep in touch with their relatives, and whose dynamic is slower than friendship [7]. In this paper we present a proposal for building and enhancing personal kinship networks by exploring and searching existing acquaintances and friendship

V. Carchiolo, V. Di Martino · A. Longheu · M. Malgeri · G. Mangioni
Dip. Ingegneria Elettrica, Elettronica e Informatica - Università degli Studi di Catania - Italy
e-mail: {Vincenza.Carchiolo,Alessandro.Longheu,Michele.Malgeri,
Giuseppe.Mangioni}@dieei.unict.it, vdimarti@gmail.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 181–186.
springerlink.com

relations that connect relatives among themselves and with other people. The proposed system (named *Kinship Search System* or *KSS*) allows each user to craft his genealogical tree using tailored search and integration of data concerning his relatives [8], also leveraging data from other genealogical trees linked via acquaintances and friendship cross relations.

In sec. 2 we provide an overview of KSS, whereas in sec. 3 we describe how the kinship network is crafted, also presenting a case study to show the effectiveness of out proposal, discussing final remarks and future works in sec. 4.

2 The Social Network: Relatives, Friends and Acquaintances

The scenario we consider concerns kinship, i.e. a social network where people establish and hold family and relatives relationships. In addition, both acquaintances and friend relationships are present in the network to model real social connections, and we leverage these links to help people in finding their relatives.

We distinguish a *User* from a *Profile*: the former is a real person that interacts with the system, whereas the latter consists of information provided by users to represent persons they know. A profile uniquely identifies a given person and contains all his *personal information* (name, nationality, birth/death places and dates, etc.), his *life events* (what, where and when they happen) and his *connections* with other profiles (mother, colleague, son, wife, friend, etc.).

KSS provide users with the following functions:

- when a user logs in for the first time, he can create his own profile from scratch, however a profile for a (currently not existing) user p can also be already defined by one of his acquaintances. If so, as soon as p will eventually join the network (becoming a user), KSS detects that an existing profile for him is present, and allows p to decide whether associate or not to that profile, thus avoiding to create duplicates. If more profiles are actually available for p, KSS collects and integrates them in order to propose p with a single profile he can associate to [1]. If p accepts the existing profile, he also inherits all links already established for that profile.
- a user can either connect to existing profiles or create new ones (in addition to his own) in order to build his genealogical tree and/or get connected with all people he knows.

Note that profiles can be duplicated, ambiguous, or some information can be missing, incomplete, wrong due to mispelling or simply outdated, therefore discovering relatives in the network can be a hard task. To deal with these situations, KSS helps users in finding the best suitable profiles of relatives they are looking for working either in off-line mode, by exploring each possible path leading to profiles of potential relatives, or on-line, providing interactive helps in the following cases:

- *automated search* - it occurs not only when the user p logs into KSS, but also whenever p creates a new profile, since other profiles for the same person might already exist. KSS searches for profiles similar to the newly created, and propose

them to p according to their relevance so p can confirm whether they represent the same people he just described. If so, KSS integrates these profiles into a single one, otherwise in the network will co-exist both the new profile and those similar.
- *on-demand search* - this may occur simply when the user searches for a specific profile/user, or when he cannot increase his family tree (i.e. no more profiles of interest have been found); in this case tailored searches can be performed on external data sources e.g. Ellis Island [2], wedding records, births records etc. and KSS will propose to the user new profiles based on these external data.

3 Crafting a Kinship Network

The goal of KSS is to help users to get connected with his family and relatives, therefore a search engine that discovers, creates, ranks and proposes a list of profiles the user may know is built.

To get them, KSS searches exploiting all user's information i.e. his profile and all those directly or indirectly connected with him (named *candidates*). Searching on a candidate allows to retrieve a set of nodes that can be potentially related with the user's profile; results are scored and the higher the score the more "suggested" a node is. Note that however wrong or missing information is always possible (the data model is indeed schema-free) and this can determine low scores even for relevant candidates.

KSS therefore executes all searches on candidates in order to achieve a global view that considers not only the score of a node, but also (1) how short the path from the starting node to the suggested ones is (the shorter a path, the more likely a node is suggested), (2) the number of candidates a node has been recommended by and (3) the type of searches that have been performed.

In particular, KSS starts searching the network for the user's profile, however he can also ask for other profiles, therefore a *work-list* of candidates is specified, (with the user's profile as the first one), along with a set of search policies. Here, a *policy* represents the common behavior for a set of search algorithms, for instance the *family* policy specifies a set of algorithms where just relatives nodes are involved during the search (policies and search algorithms are independent each other). The set of searching algorithms selected according to chosen policies is then executed on the candidate.

The execution can have two effects: (1) providing suggestions by collecting and scoring nodes or (2) expanding the search by adding new candidates into the work-list. Therefore the work-list can grow up while its elements are considered and, in a generic execution scenario, there can be more than one candidate in the work-list and more than one algorithm specified for each of them. After all algorithms for candidates have been executed, results are collected; this happens at three levels:

1. **candidate level**, where each candidate collects all results coming from associated algorithms. Note that a profile can occur more than once, e.g. one can be a user's relative, but also one of his colleagues; in this case, scores get unified

2. **manager level**: results coming from all candidates are collected; if a profile occurs more than once, they are merged into one resulting object having all the paths and relative scores representing its multiple occurrences;
3. **engine level**, where only profiles with a relevant score (according to a proper threshold) are suggested

Once all profiles have been collected, they must be compared in order establish a kinship degree; to this purpose, any algorithm:

1. queries the graph searching for every relevant node (according to policies) the result is a list of set of nodes, each set containing profiles related to the candidate according to some properties.
2. iterates over all these sets to determine weights (based on statistical indexes) used for nodes comparison.
3. compares the candidate node with all of the nodes in the initial set. In case of match between two profiles [10], the relative weight is extracted and used to compute the final kinship score for that node.

3.1 A Case Study: Perfect Match

As discussed previously, several algorithms can be plugged into KSS As a case study, we show the *Perfect Match* algorithm, used each time a new profile (`P`) is created to check the existence of similar nodes. The algorithm creates a list of nodes related to `P` ordered by similarity. It queries the graph exploiting the information of `P`, disregarding all nodes the `P`'s profile is already connected to, and it extracts the initial data-set used to assess statistical indexes. For each node belonging to the initial set, KSS starts collecting statistics iterating over all the properties they have in common with `P` and counts how many different values exist for every property. As shown in Table 1, the weight assigned to each property is calculated according to the inverse of the number of nodes that provide that property.

Table 1 Example of property weights assessment for a given a set of nodes

Property	Node1	Node2	Node3	Node4	Weight
Name	John	Jim	Jake	Jack	1
Surname	Smith	Smith	Smith	Smith	0.25
Nationality	Italian	USA	-	USA	0.66

Once all statistic indexes are collected, `P` is compared against all nodes in the initial set using weights to assess their similarity. For each `P`'s property, event or connection, the algorithm checks whether the same property key is present on the candidate node and compare them properly assigning a score that represent their similarity, adopting de-facto a VSM-based approach [3] (details are here omitted [1].

3.2 Results

Data used for this case study were extracted from the public italian historical birth archive using data ranging from 1800 to 1900s and within a specific geographic area.

The KSS data archive has been populated with records containing information about a person's and his parents' (e.g. name, surname, birth date, etc.), then we obtained three profile nodes containing the proper attributes and representing the person and his parents linked with the right edge type (father, mother or son-daughter). Finally, the data-set contains about 187 000 nodes and 310 000 edges.

First experiment aims at searching corresponding profiles in order to prune redundant information. Therefore, we applied the perfect match algorithm to the above dataset and we merged all those profiles with matching values greater than a given threshold. We expect that merging two nodes has two main effects, (1) the profile number in the network should reduce, and (2) the number of isolated graphs become smaller, since families get joined together. The results of applying the algorithm over the network three times in a row are shown in figure 1, where threshold is 80%.

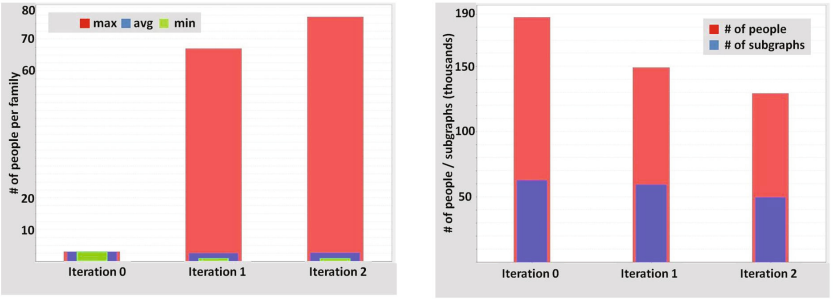


Fig. 1 (a) Family size and (b) Population and Subgraph sizes vs algoritmh iterations

In figure 1 (a) we show the maximum, minimum and average size of a family. From the initial situation of three members in each family (by construction) the families size grow up that highlight the effectiveness of proposed methods to create the kinship. Let us note that the effect emerges soon (first iteration) The chart in figure 1 (b) represents the number of profiles and the number of isolated graphs (a connected family) in the network, with respect to the number of iteration. Both numbers decrease and this decrease is more evident in the first iteration than in the next ones. This depends on the fact that, after the first iteration, all matching nodes were merged and few good matches should have been left for the next iterations. On the contrary, all the information added when merging profiles during the first iteration, allowed the algorithm to find new good matches that were ignored at first instance for lack of information. Also the number of isolated subgraphs reduce considerably, improving overall graph connectivity and quality.

4 Conclusions

In this paper we presented a proposal for building and enhancing kinship networks, showing the effectiveness and expandability of our approach. Further works may include (1) extending the set of search criteria, (2) performing comparative tests with different data sets, search criteria and algorithms, in order to assess how these three factor actually affect performances.

References

1. Di Martino, V.: Emerging relations in a property multi-graph: the case study of parental networks, DIEEI Technical Report (2012)
2. American Family Immigration History Center, AFIHC (2001), http://www.ellisisland.org/
3. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM (November 1975)
4. Bargh, J.A., McKenna, K.Y.A.: The Internet and Social Life. Annual Review of Psychology (2004)
5. Nie, N.H.: Sociability, Interpersonal Relations, and the Internet: Reconciling Conflicting Findings (2001)
6. Scott, J.: Social Network Analysis: A Handbook, 2nd edn. Sage Publications, London (2000)
7. Roberts, S.G.B., Dunbar, R.I.M.: Communication in social networks: Effects of kinship, network size, and emotional closeness. Personal Relationships. Blackwell Publishing Ltd. (2011)
8. Yu, B., Singh, M.P.: Searching social networks. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems. ACM, Melbourne (2003)
9. Newman, M.: The structure and function of complex networks. SIAM Review 45 (2003)
10. Lenzerini, M.: Data integration: a theoretical perspective. In: Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. ACM (2002)

The Effects of Pre-trusted Peers Misbehaviour on EigenTrust

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

Abstract. Trust is a widespread mechanism to establish and conduct successful social collaborations, expecially in today's Internet, where people perform more and more activities on-line, from social networking to e-learning, e-commerce and others. Several algorithms are available for trust assessment, such as the well-known EigenTrust to PowerTrust, GossipTrust, TrustWebRank to cite some. An evaluation of such algorithms must take into account several factors, e.g. performances, scalability and robustness against malicious attacks, which plays a key role in ensuring algorithm's effectiveness. In this paper we study the effects of malicious peers on the EigenTrust algorithm, focusing on the (mis)behaviour of its *pre-trusted* peers, using a proper simulator to conduct several experiments under different conditions. Results show that EigenTrust effectiveness may be significantly affected by inactive and/or malicious pre-trusted peers.

1 Introduction

Today's Internet moved from a content-centric to a people-centric vision, where people perform more and more activities on-line, from social networking to e-learning, Peer-to-Peer (P2P) networking, e-commerce and so on. Several of these applications leverage trust to establish and conduct successful social collaborations [1]. Trust models indeed play an important role in ensuring the security and the reliability of interactions in these open distributed environments where the assessment of each entity's trust generally depends on his past experiences as well as on recommendation information coming from other entities.

How trust is actually established between two entities (users, peers, agents or whatever) is subject to many interpretations, from psychology-based (where trust is based on feeling or empathy) to more unbiased approaches, as it is outlined in

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

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 187–197.
springerlink.com

the work by Marsh [2], where the first analytical formulation of trust relationship between two persons was given. In McKnight [3] a taxonomy of motivations and aspect in trust assessments can be found, whereas the work by Artz and Gil [4] offers a complete overview of recent issues about trust.

Several algorithms are available for trust assessment, from the well-known EigenTrust [5], which associates to each node a global trust index using an algorithm inspired by PageRank [6, 7] and by the random walker [8] mathematical formalization, to others as PowerTrust [9], that provides scalability for P2P networks, GossipTrust [10], that enables fast reputation aggregation in trust evaluation, and the recent TrustWebRank [11], where personalization and dynamics of trust are addressed.

When evaluating such algorithms, some factors should be considered:

- the performances related to the implementation overhead at each agent/peer, to the speed in reputation aggregation, to the accuracy and dissemination of trust values and so on
- the scalability, expecially for very large networks as it occurs in the real world
- the robustness against attacks from malicious peers and misbehaviour of free-riders
- the dependability on the type of network (e.g. scale-free, random)

Among these, the robustness is one of the most relevant since it can significantly affect the effectiveness of any approach in real context, where misbehaviours always represent a threat.

Starting from these considerations, in this paper we analyze the effects of malicious peers on the trust assessment of the EigenTrust algorithm. In particular, we focus on the behaviour of some critical nodes, called *pre-trusted* peers, a seed of trusted accounts that hold an important role in routing information and have a strong influence on the quality of the algorithm [5].

The study of pre-trusted peers misbehaviour requires to consider several scenarios. More generally, any attempt to evaluate a trust metric actually requires many experiments to be conducted on real networks under different conditions; unfortunately, this is not feasible for several reasons, for instance the changeability of networks in terms of topology, number of peers and/or links, or the fact that the (correct) behaviour cannot be super-imposed on users/agents, thus determining unwanted or even unpredictable results.

These simple yet not exhaustive considerations led us to develop a simulator for fully-customizable P2P networks that supports a modular reputation-based trust system, i.e. where both the trust metric and the scenario can be tailored to any needs.

We use EigenTrust into our simulator analyzing the attack scenarios when of one or more pre-trusted peers go offline and/or become malicious, showing how the effectiveness of EigenTrust is affected by changing the behaviour of such peers.

The paper is organized as follows: section 2 recalls the Eigentrust algorithm. The simulator is presented in section 3 whereas its use and results concerning Eigentrust are shown in section 4. Finally, section 5 presents conclusions and further works.

2 The EigenTrust algorithm

In the following we briefly recall EigenTrust, a reputation-based trust algorithm that assigns each peer i an unique global trust value using the experiences of other peers with i.

In EigenTrust, the evaluation is performed by a peer i by storing the number of successful and unsuccessful transactions he had with another peer j, where a transaction can be for instance a purchase in an e-commerce website or a file download in a P2P network. The algebraic sum of total number of transactions s_{ij} is used to build a normalized local trust values i assigns to j, defined as in eq. 1:

$$c_{ij} = \frac{max(s_{ij}, 0)}{\sum_j max(s_{ij}, 0)} \tag{1}$$

The c_{ij} can be evaluated only if i directly interacted with j in the past. To assess *indirect* trust, a peer i can ask its neighbors (in EigenTrust they are known as *acquaintances*) and, if they know the peer k, the indirect trust between i and k is calculated as in eq. 2, where i mediates the direct opinion of his acquaintances (c_{jk}) with his opinions about them (c_{ij}). In eq. 2 this is also expressed in matrix notation where t_i is the set of trust values i provides to all nodes k in the network.

$$t_{ik} = \sum_j c_{ij} \cdot c_{jk} \qquad t_i = C^T \cdot c_i \tag{2}$$

In general, the process of asking acquaintances to get their opinion could require n steps along the network (until direct trust is achieved), hence the equation 2 is generalized as in equation 3:

$$t_i^{(n)} = C^T \cdot t_i^{(n-1)} \qquad \text{or, equivalently:} \qquad t_i^{(n)} = (C^T)^n \cdot t_i^{(0)} \tag{3}$$

where $t_i^{(0)} = c_{ij}$. As deeply discussed in [5], if the matrix C is irreducible and aperiodic, for n sufficiently large the vector $t_i^{(n)}$ will converge to a value t that is the same for any peer i; this vector is the set of *global* trust values of all nodes in the network. From an analytical point of view, this means that the iterative eq. 3 becomes the eq. 4, where it is trivial to note that t is the left main eigenvector of C:

$$t = C^T \cdot t \tag{4}$$

To actually guarantee the irreducibility and aperiodicity of C, some considerations are needed, leading to slightly modified versions of eq. 1 and 3 (see [5] for more details). EigenTrust authors also offer a probabilistic interpretation of their algorithm, based on the random walker model [8, 12], in particular the normalized local trust c_{ij} can be interpreted as the probability for a random walker to jump from i to j; at a steady state, the eq. 4 says that a random walker is more likely to be at reputable peers than unreputable (the more is the probability, the more is the trust). According to this interpretation, the normalized local trust matrix C in eq. 4 is a Markov chain, whose stationary distribution is the global trust vector t.

3 The Simulator for Trust Assessment

As discussed in the introduction, testing properties and performances of P2P trust-based networks may be a difficult and time-consuming task, so we have implemented a simulator written in C++ to hold full control on all phenomena occurring in the network. Indeed, it allows to setup a fully-customizable P2P network, in particular by modeling and managing resources upload/download relationships and by applying different metric for trust assessment, also allowing to simulate different attack scenarios; for instance in our experiments we addressed the case when one or more pre-trusted peers go offline, and/or become malicious. Some of simulator features are described in the following.

The choice of distributed algorithms for P2P networks simulations is often affected by peers interconnections and by traffic shapes, hence it is hard to recreate the same execution environment during different simulations. An alternative is represented by centralized softwares which hold all the network parameters; since we are not specifically dealing with distributed aspects (e.g. network traffic analysis), our simulator follows a centralized approach. Moreover, in distributed P2P algorithms security is crucial since packets traverse the network and for instance a malicious peer could easily change the packets content before forwarding; again, since we are not interested in any security related issue, authentication and integrity are out of the scope of this work.

The structure of our simulator allows the P2P network to be simulated separately from the trust assessment, so it is possible to guarantee the same network conditions for each simulation and to test different trust metrics and/or different kinds of attacks with the same network model.

The simulator is split into three main modules, i.e. (1) a generator of the content distribution, (2) a query executor and (3) a trust evaluator, described in the following subsections, and it allows full customization of the P2P network according to the parameters described in table 1.

Table 1 Simulator Parameters

Parameter	Description
Npeer	# of peers in the network, excluded malicious
Threatmodel	Indicates the attack model
Mode	The download source selection: 1) deterministic, 2) probabilistic
Numquery	# of queries for each cycle
Numquerycycle	# of iterations
numOfCategory	Total # of files categories
minCategory	Minimum # of categories in which a peer is interested in
percentUpTimeProcessQuery	Percent of time in which a peer processes queries
percentUpTimeIssuesQuery	Percent of time in which a peer issues queries
numPreMal	# of pre-trusted peers that become malicious
preMalIteration	The iteration in which pre-trusted peers become malicious (0 if from the beginning)
numInactive	# of pre-trusted peers that become inactive
inactiveIteration	The iteration in which pre-trusted peers become inactive

3.1 Content Distribution Generator

P2P networks rely on the voluntary participation of peers in sharing resources, thus joining and leaving of peers can be frequent, making such networks a highly dynamic system. To provide an effective service, a proper distribution of files among peers should be adopted in order to guarantee enough resources replication. The number of replicated files indeed increases the performance and the robustness of the network; whenever one of the file owners' is inactive or down, the same file is provided by someone else. The Content Distribution Generator (CDG) aims at representing a real resource distribution in the simulated P2P network; this is achieved by acting on the total number of shared files and number of replica for each file.

CDG leverages existing measurement studies in P2P file sharing systems to achieve its goal [13]. In particular, the volume and the variety of the data are characteristics that determine the dynamic of the P2P network. Indeed, since in a common P2P network a peer can decide to share or not files, if a large amount of peers decide to share a large volume of data, then the dynamic of the network is fast, i.e. queries get answers more rapidly. Conversely, if only a few peers decide to share files, there will not be enough replicas, and the dynamic of the network will be limited. In summary, a P2P network is highly dependent from how many files its peers are going to share. A reputation system can be used as an effective incentive to share resources both for free riders and for all those peers sharing a few resources; this can be achieved for instance by providing peers with an available bandwidth proportional to their trust value. In the simulator a specific number of files is assigned to each peer. The total number of files shared by a peer i is denoted by Fi.

In real networks, it has been noticed [14] that a peer is interested in a specific kinds of files; this implies that a peer is going to share files within a limited set of categories. Moreover, files are characterized by different popularity, hence there are files replicated in a large numbers of copies, and others that are not replicated at all. Files are then arranged into categories, and in each category they are distributed with different popularity. Therefore, when a peer joins the network, it chooses to be interested in a certain number of categories and to share files only in these categories.

In the simulator, the popularity of categories and the popularity of files are represented with the well-known Zpif distributions [15]. The number of categories by default is twenty, and the number of categories in which a peer is interested is three; however both parameters can be customized. The choice of the categories follows a Zpif distribution so popular categories will have more chance to be chosen. In our model, files are recognized by their category and their popularity (rank). A file is uniquely identified by the symbol $f_{c,r}$ which represents the file in category c and rank r, unique within a category. In general, with n content categories it is possible to define the set of categories $C = c1, c2, ..., cn$, in which c_i is the rank of the category i. The popularity is represented in ascending order, so if $i > j$, than $r_i > r_j$. All of these informations are stored inside each peer. Each peer holds a list of his files shared on the network, the categories they belong to, and the number of files

per category. Finally, a centralized server maintains all this informations by storing a list of peer objects.

The main goal of CDG is to issue random queries. The number of generated queries depends on the number of queries per simulation cycle, and the number of query cycles (both customizable). For each query, a random peer issuer is selected, and the query is propagated along the network. In our query model a pre-trusted peer has 100% of his uptime in issuing query, whereas a standard peer has a certain probability to be offline, so it will lose the possibility of issuing query. This probability has a uniform distribution over [0%, 50%]. The next choice concerns the file to request in the query. Content category and files are selected randomly by a roulette-wheel selection based on their popularity. Therefore, the list of peers is scanned to check which peers have the requested file. As in the issuing query phase, pre-trusted peers always answer to a query, whereas standard peers could be offline with a given probability. The effect of the presence of malicious peers is modeled connecting them to the most trusted peers in the network. In this way they will answer at least to the 20% of the queries. If no one answers to the query, it will be dropped.

3.2 Queries Simulator

The steps QS follows are described in the following. After the network population is set in terms of number of total peers, and number of pre-trusted and malicious peers, the simulator loads simulation parameters.

Then, the query file generated by CDG is read and the execution of the queries starts.

The Queries Simulator module (QS) uses a query cycle model to simulate the evolution of the network in terms of upload/download relations. This is performed by allowing the selection of the downloading source with different metrics. In each cycle a peer may be active or inactive and may issue a query or not. If a peer issues a query, it will wait for a response, then choosing a downloading source, according to a deterministic or probabilistic model.

The process ends with the feedback assignment about the resource downloaded. In particular, the feedback is assigned in two different ways, depending on whether the requested peer is malicious or not. Good peers assign positive feedback if the downloaded file is authentic; malicious peers instead assign a positive feedback if the downloaded file is inauthentic.

A query cycle finishes when the numbers of queries per cycle is reached, and all peers who have issued a query have downloaded the resource. At this stage it will be possible to collect information about the network evolution and traffic/resource statistics. At the end of each query cycle the simulator also performs the evaluation of the global value trust and updates the new trust vector which is used in the next iteration. All this information is available via text output files generated by the simulator.

3.3 *Trust Assessment*

Trust Assessment (TA) is the third module of our simulator, whose goal is to evaluate trust as a global (single) numerical value for each peer, according to the EigenTrust algorithm. The assessment can be easily parametrized, an example of the configuration file is:

```
GENERATION
SIMULATION
output=twb.csv
epsilon=0.01
metric=Eigentrust
exp=10
percMal=10
```

The keyword GENERATION launches a new CDG, but it is also possible to generate content distribution just once and then launch TA-related experiments with different parameters. The keyword SIMULATION launches a new simulation; the adjustable parameters concern the percentage of error in evaluating the algorithm converge condition (*epsilon*), the trust algorithm used (*metric*), the number of the experiments (*exp*), and the percentage of malicious peers (*percMal*). The percentage of malicious peers can be set to a specific value or it could be 0. When set to 0 ,the trust assessment launches several experiments with different percentages of malicious ranging from 0% to 70% with step of 10%. This is the classical use case used in our experiments.

4 Results

We performed several experiments in order to evaluate how the presence of both inactive and malicious pre-trusted peers affects the EigenTrust algorithm. The experiments have been performed on a typical P2P network that consists of honest peers, which are interested in uploading and downloading resources, and malicious peers which are interested in sharing inauthentic files trying to subvert the reputation system.

At the end of each simulation, we collected statistics about the evolution of the P2P network. The main result of each experiment focuses on the comparison between authentic and inauthentic files downloaded. To avoid the problem of introducing biases, all the experiments have been performed several times with the same condition, changing the seed used to generate random numbers, and the results are averaged. Note that the statistics are collected since the P2P network reach a steady state (not before).

The first set of experiments concerns with the difference in downloading sources selection (1). This can be performed with two different modalities: deterministic and probabilistic. The experiments simulate the dynamic of a small network with 20 peers, showing the percentage of the provided resources of each peer. The P2P network only contains honest peers and the experiment is performed 10 times averaging the results.

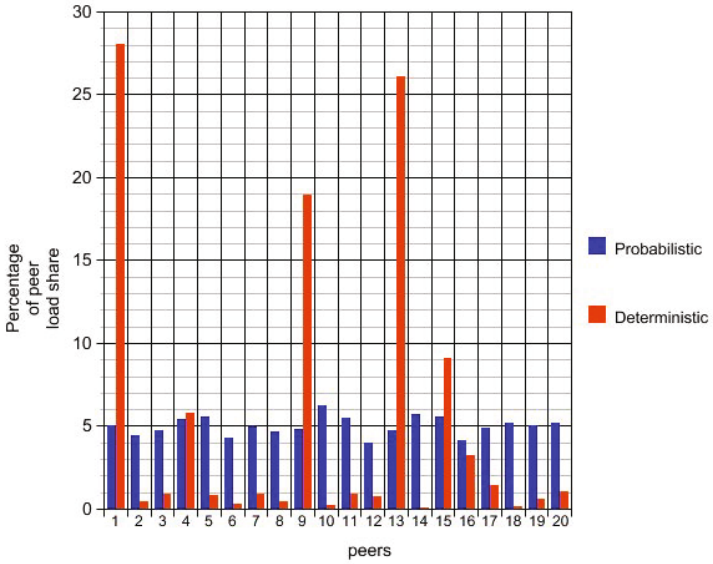


Fig. 1 Load distribution on a P2P network using deterministic and probabilistic downloading source selection

The load distribution is depicted as the fraction of files provided from each peer, divided by the total number of files provided in the network. Figure 1 shows that with a deterministic selection of the downloading source the load distribution is held just by a few peers, that will always be chosen if they answer to a query. This leads to a system where few peers are overloaded by requests, and many peers are able to provide only unique resources that popular peers do not have. In this context, there is a monopoly of trust, which is high just for very popular peers. Conversely, in the probabilistic model the traffic is not based on a trust metric and it is almost equally distributed among all peers; note that these results agree with those presented in the original EigenTrust paper [5]. The use of a probabilistic model avoids overloading peers with continuous requests, but in a P2P network with malicious peers this could increase the number of inauthentic files downloaded. The solution adopted use a probabilistic model based on a reputation-based trust algorithm (EigenTrust in our case), in particular the downloading source is selected according to a probability proportional to its trust.

The next experiments concern the number of authentic and inauthentic files downloaded, and they have been performed in a P2P network with 63 honest peers. Table 2 summarizes the overall settings. To allow newcomers to build their reputation, there is a 10% of probability that a peer with a zero trust value is chosen. This also introduces a 10% of probability to choose a malicious peer hence to receive an inauthentic file. Malicious peers are connected to the most trustworthy peers and they answer to the 20% of queries. The simulation also takes into account the

possibility that a honest peer provides an inauthentic file; this is possible because honest peers may have uncorreclty categorized the file, or because they did not delete an inauthentic file (downloaded in the past) from the shared folder.

Table 2 Parameters of the experiments

Network	# of honest peers	60
	# of malicious peers	[0%,70%] of honest peers
	# of pre-trusted peers	3
Content Distribution	# of categories	20
	# of distinct files in a peer	Provided by CDG
	# of distinct files of a peer i in category j	Uniform random distribution over peer is total number of distinct files
	% of queries malicious peers answer to	20 %
	% of queries pre-trusted peers answer to	5 %
	% of up-time honest peers spend in processing queries	Uniform random distribution in the range [0%, 100%]
	% of up-time honest peers spend in issuing queries	Uniform random distribution in the range [0%, 50%]
	% of up-time pre-trusted peers spend in processing queries	1
	% of up-time pre-trusted peers spend in issuing queries	1
Peer Behavior	% of download requests in which honest peer i returns inauthentic file	5%
	% of downloads requests in which malicious peers return authentic files	0 %
	Downloading source selection	Probabilistic
	Probability that an untrusted peer (trust=zero) is selected as download source	10%
Simulation	# of simulation cycles	30
	# of query cycles in one simulation cycle	50
	# of experiments	50

The simulator we developed has been tested considering the same attack strategies presented in EigenTrust (i.e. threat models named 'A' and 'B'; please refer to [5] for more details), and the results we obtained are the same as EigenTrust authors obtained, thus validating our simulation tool.

In this work we show that effectiveness of the EigenTrust algorithm is affected by changing the behaviour of pre-trusting peers, due to their crucial role in trust assessment [5].

The first case represents the case when pre-trusted peers are always up in answering queries. This behaviour is changed assuming that after a certain number of iterations (customizable in simulator's configuration files), a pre-trusted peer becomes inactive. The parameters of the network are the same as table 2, except for the number of experiments which is set to 100. The results are related to the scenario where 70% of malicious peers are present, and shows the trend of the percentage of inauthentic files downloaded as the number of inactive pre-trusted peers changes. Figure 2 shows the decay in the performance of the algorithm due to the fact that the component able to break a community of malicious peers is missing.

The final experiments have been performed to test EigenTrust with malicious pre-trusted peers. These experiments are shown in figure 3. The parameters of the system have been used in accordance with those in table 2. The percentage of malicious peers is 70%, and the downloading source selection model is probabilistic.

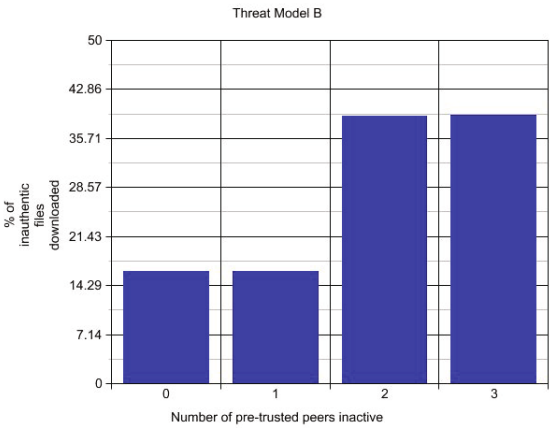


Fig. 2 % of inauthentic downloads vs # of inactive pre-trusted peers (threat model B described in [5], 70% of malicious peers)

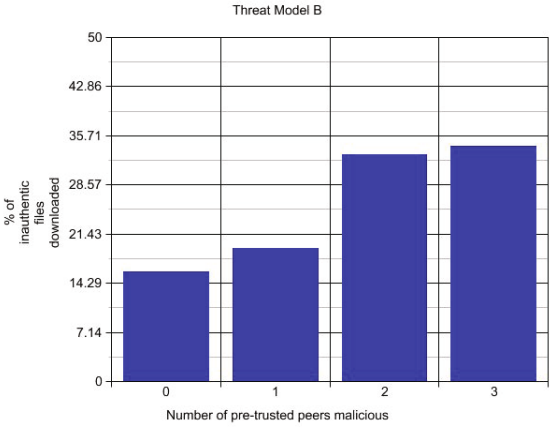


Fig. 3 % of inauthentic downloads vs # of malicious pre-trusted peers (threat model B described in [5], 70% of malicious peers)

The experiments have been performed varying the number of pre-trusted peers that in a specific iteration become malicious. As shown in figure 3 the performance of the algorithm decays; this is what happened before a steady state is achieved, when pre-trusted peer global trust value were still high. After transient states, the network is able to isolate these peers reducing the number of inauthentic download, even if not as significantly as in standard conditions.

5 Conclusions

This paper introduced the influence of inactive and malicious peers in a trust-based P2P network. In particular, we considered the well-known EigenTrust algorithm showing how its effectiveness is affected by misbehaviour of its pre-trusted peers.

Some issues deserve further attention, in particular:

- thanks to its modular nature, the simulator can be used to assess the attack robustness of other metrics in addition to EigenTrust;
- moreover, using the same simulation conditions we can also provide a reasonable comparison among different metrics;
- other properties in addition to the attack robustness should also be investigated, as performance and scalability; the simulator can help in addressing such issue

We finally thanks Andrea Longo and Francesco Munzone for their support during the development of the simulator.

References

1. Scott, J.: Social Networks Analysis: A Handbook. Sage Pubblications, London (2000)
2. Marsh, S.: Formalising trust as a computational concept. Technical report, University of Stirling, PhD thesis (1994)
3. McKnight, D.H., Chervany, N.L.: The meanings of trust. Technical report, Minneapolis, MN - USA (1996)
4. Artz, D., Gil, Y.: A survey of trust in computer science and the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 58–71 (2007)
5. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the Twelfth International World Wide Web Conference (2003)
6. Berkhin, P.: A survey on pagerank computing. Internet Mathematics 2(1), 73–120 (2005)
7. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab (1999)
8. Pearson, K.: The problem of the random walk. Nature 72, 294–294 (1905)
9. Zhou, R., Hwang, K.: Powertrust: A robust and scalable reputation system for trusted peer-to-peer computing. IEEE Trans. Parallel Distrib. Syst. 18(4), 460–473 (2007)
10. Zhou, R., Hwang, K., Cai, M.: Gossiptrust for fast reputation aggregation in peer-to-peer networks. IEEE Trans. on Knowl. and Data Eng. 20(9), 1282–1295 (2008)
11. Walter, F.E., Battiston, S., Schweitzer, F.: Personalised and dynamic trust in social networks. In: Bergman, L.D., Tuzhilin, A., Burke, R.D., Felfernig, A., Schmidt-Thieme, L. (eds.) RecSys, pp. 197–204. ACM (2009)
12. Hughes, B.D.: Random Walks and Random Environments. Oxford Univ. Press, USA (1995)
13. Saroiu, S., Gummadi, P.K., Gribble, S.D.: A measurement study of peer-to-peer file sharing systems (2002)
14. Gummadi, P.K., Dunn, R.J., Saroiu, S., Gribble, S.D., Levy, H.M., Zahorjan, J.: Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In: SOSP, pp. 314–329 (2003)
15. Zipf, G.K.: Human behavior and the principle of least effort. Journal of Clinical Psychology 6(3), 306 (1950)

Sensorization and Intelligent Systems in Energetic Sustainable Environments

Fábio Silva, David Cuevas, Cesar Analide, José Neves, and José Marques

Abstract. Sustainability is an important topic of discussion in our world. However, measuring sustainability and assessing behaviors is not always easy. Indeed, and in order to fulfill this goal, in this work it will be proposed a multi-agent based architecture to measure and assess sustainable indicators taken from a given environment. These evaluations will be based on past and present behaviors of the users and the particularities of the setting, leading to the evaluation of workable indicators such as gas emissions, energetic consumption and the users fitting with respect to the milieu. Special attention is given to user interaction and user attributes to calculate sustainable indicators for each type of structure, i.e., the aim of this scheme is to promote sustainability awareness and sustainable actions through the use of sustainable markers calculated in terms of the information gathered from the environment.

1 Introduction

Ambient Intelligence (AmI) is still considered an emergent technology that may be embedded into environments, making them both sensitive and responsive. In this sense AmI may be used to achieve several objectives inside such environments, e.g., in sustainability assessment, enforcement and suggestion [5]. In fact, there is an increasing source of concern as more researchers make use of computational resources to find sustainable equilibriums. Sustainable models developed in the literature are also focus of research and improvement, although they may differ on their approach. Some of these models use economical metrics in order to assess sustainability, while others follow social and environmental perspectives in a more accurate form [11]. The work presented uses a multi-agent system to obtain information about an environment so that deliberative and reactive decisions concerning sustainability can be made. Indeed, multi-agent system architectures that foresee

Fábio Silva · David Cuevas · Cesar Analide · José Neves · José Marques
Department of Informatics, University of Minho
e-mail: {f.aandree,davidjfcuevas,josealbertomarques}@gmail.com,
{analide,jneves}@di.uminho.pt

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 199–204.
springerlink.com

these problems, may be found in [9]. Performance tests showed good results for the service discovery in terms of flexibility and interoperability. Decisions are created using reasoning processes upon the data gathered either with machine learning or context-aware computing, so that specialized intelligent decisions may be made for each user present in an AmI setting. With respect to this area of research, Machine Learning inside AmI environments have already been used for feats such as human activity tracking [3]. Current sustainability assessment considers different indicators and sub-models created and used by specialized people. Our work aims to make it simple to assess and determine sustainability indicators in an intelligent environment through the use of multi-agent architecture and environment sensing. It has been demonstrated by previous research that when people are aware of the consequences of their actions in detail, and they are set with an objective, they tend to act in the best manner to attain it, as it was the case with eletrical consumption [4]. It is expected that the use of architectures like the one proposed in this paper causes the same effect on sustainable measures and user behaviour as it was demonstrated for electric consumption and user behaviour.

2 Previous Study

In this section there are presented studies related to the research being conducted. As such each category will be presented and a general descritpion will be made.

Sustainability is a subject of concern for the assurance of the steadiness, viability and use of a system. Currently different approaches to measure and assess sustainability were proposed in the literature, with some focusing on an economical perception, while others emphasize on environment or social perspectives [11]. Sustainability indicators have been useful at pointing out unsustainable practices, however, they are not as good to define and guarantee sustainability [7]. A common accepted definition for the notion of sustainability concerns an equilibrium from social, economical and environmental factors. When some of these features cannot be met, the system is not considered sustainable, but it may be pondered viable, bearable or equitable [11]. In contexts like intelligent buildings, there are commitments to build Key Performance Indicators (KPIs) to monitor sustainability, and act as sustainability indicators from information gathered by a panel of experts [2].

Machine Learning (ML) techniques or methods concede the modeling and learning of preferences and habits in different contexts. These methods also sanction the acquiring of past and current trends and predict future results. From information assembled from different environments, ML techniques may derive models of behaviour and interaction based on specialized backgrounds (e.g., users, environment, social interaction or consumption). ML and Data Mining (DM) techniques may also be used to obtain information about user's habits in AmI settings, from data gathered by sensors in the environment, namely using practices such as Sequence Discovery [3], Fuzzy Logic [6], Genetic Programming, Multi-Layer Perceptron, Evolutionary Intelligence [8] or combinations of these techniques [12].

Context-Aware Computing is a component of a ubiquitous computing computing environment. One hopes to make personalized decisions based on contextual factors that may lead to different results to the same situation, once placed in different contexts [10]. Evidence of these context-aware systems can be found in e-Commerce, Information Retrieval and Ubiquitous Systems. In these examples context awareness is used to personalize systems and applications to the present user context [1]. Despite the interest in context-aware computing, context definition for applications and systems is still challenging in some situations.

3 Multi-Agent System for Sustainable Environments

This case study presented encompasses an intelligent environment and its user interactions in order to obtain information about its current sustainable condition. The source of such interest may be economical challenges or a conscious mind about the future sustainability of his environment, as described in [4]. The user is supposed to do daily routines such as cooking, sleeping or watching television. Nevertheless, the system records each action on the part of the user interactions, and uses such information to reason about existing sustainability indicators and metrics of control concerning the environment i. e., resource utilization, CO_2 emissions and of its own attendance on each internal premise in the environment for example.

The objective is to use different model representations, for example, environmental vairiables, ambiental variables, interactions and appliance consumptions. These models may receive input from different sensors distributed in the environment or mathematical formulae. There may also be the case where a different intrepretation of the values of a primary model or group of models may derive new ones, i.e., electric costs and carbon emission from electric consumption. Therefore, the user is aware of his impact on his environment.

System Architecture as proposed is a multi-agent system which encompasses a set of agents accountable for the assessment of current sustainability indicators in the environment. These agents communicate with the environment in order to retrieve data about it and use intelligent models to assess it based on sustainability indicators. Indeed, the proposed architecture uses 4 (four) different types of agents, namely:

- Sensing - an agent connected to sensors that is responsible to gather data about the environment, such as user presence or energetic consumption;
- Modeling - an agent responsible to model a representation of the essential aspects of a system which presents knowledge and relies on both mathematical and physical formulas as well as data from the sensing agents;
- Reasoning - an agent that reasons about perspectives, setting a method that fulfills the formal requirements for a theory of context, and offers an explanatory account of contextual reasoning in terms of information flow;
- Actuator - an agent that acts on the environment and monitors the execution of those acts to reassure a correct behaviour on the system.

With this setup the environment has all the necessary agents to gather data, transform and process it into agent's knowledge models.

Sustainability Indicators available in Table 1 are in the form of the three key sustainability categories: economical metrics that include the running costs associated to the environment continuance versus the available income in the same period; social metrics that relate the presence and the possible interaction among the users that populated the environment; and environmental that assesses the the amount of carbon emissions that are generated. The values from the proposed indicator are interpreted as unsustainable if the value is below 1 or sustainable otherwise.

Table 1 Sustainability indicators

Economical	Environmental	Social
$\frac{EnergeticCosts}{IncomeAvailable}$	$\frac{EmissionTreated}{TotalEmission}$	$\frac{TimeInside}{TimeOutside}$

The overall sustainability of the environment is calculated by 1, which denotes a compromise among the three main categories of sustainability indicators.

$$S_{index} = \alpha \times I_{economic} + \beta \times I_{environmental} + \gamma \times I_{social} \quad (1)$$

The equation presented in 1 has three qualifiers, one for each sustainability indicator. The values of such indicators are between 0 and 1 where their addition totals 1. All indicators are calculated either locally or for the entire setting which sums the assessment of all premisses. This way, if the environment is considered sustainable, the user may still assess changes in premises with low supportable standards.

4 Tests and Results

In order to set up initial scenarios both a simulation and a real environment were studied. On both environments it is possible to record consumptions, presence and emissions. User notification is made through a dedicated interface that displays environment configuration, its sensors, and the levels of consumptions and emissions.

For the simulated environment, a model of a traditional home with a bedroom, a living room, a bathroom, a kitchen and a hall was conceived. The actions simulated included user movement and the swapping of the state of the appliances (on or off). Electrical consumption, user presence and carbon emissions were modelled from simulated sensor values for a period of 3 days and a total of 201 actions simulated representing a user presence of about 58% in the environment each day.

The Intelligent Systems Laboratory (ISLAB) at University of Minho was selected as the real environment. ISLAB is made of one room available to researchers where they can gather and work together. This room was modeled containing a set of

appliances such as computers and lights. In this room it was also installed sensors which recorded user presence, luminance and temperature for the period of 5 days where 2 of those were holidays. The sensing and modeling agents were used to obtain model representation of user occupancy, lightning and temperature on the ISLAB.

Table 2 Results from simulation

ISLAB Environment			
Room	Carbon emission (g)	Energetic Consumption (kW)	Presence (%)
ISLAB	7078.46	21.80	11
Home Environment			
Room	Carbon emission (g)	Energetic Consumption (kW)	Presence (%)
Bedroom	5.51	0.02	55
Kitchen	3537.32	9.95	9
Living Room	994.43	2.80	32
Bathroom	3.47	0.01	1
Hall	13.87	0.04	3

Results from both simulations are presented in Table 2 according to the formulas presented in Table 1 and data simulated and obtained from sensors. In order to estimate an economic indicator, the user is supposed to have 1,5 euros per day for daily energy consumption. From these results it is possible to calculate the sustainable indexes defined in 1 which are 3.19, 0.67 and 1.38, in the home setup, and 1.92, 0.67 and 0.14, in the ISLAB setup, for the economic, environmental and social indexes. It is also demonstrated, in the home environment, that it is economically viable, however, there is a penalization on the environmental indicator. This points out that the environment under study does not treat effciently emissions derived from its services. It is a potential source of improvement.

On the ISLAB results it is possible to demonstrate that if there is a budget of 1 euro for electricity on a daily basis then, sustainability indicators show that there is a penality not only in the environmental indicator but also in the presence indicator due to the limited time the room is occupied and its eletrical consumption. Changing the environment generates different simulation results, so simulation can be used to assess the impact of desired changes in the environment in terms of sustainability.

5 Conclusion

The proposed work features the perception of a multi-agent system able to handle sustainable monitoring and assessment in an intelligent environment. Furthermore, the use of formal models, was found to be a bright way to test and assess its sustainable actions and behaviours. It was found that it is possible to derive conclusions from the use of such systems in order to improve its efficiency and sustainability indicators. This is achieved with user awareness about his current situation, indicating which of his actions are more sustainable. As future work there is the need to further validate the model using real sensors in real world settings and design new metrics

for sustainability indicators. The multi-agent system needs to be extended with the ability to monitor different environments in parallel, and explore connections and influences among them, such as a home and an office in the same environment. A recommender system may help in the improvement of the sustainability indicators.

Acknowledgements. The research presented is partially supported by a portuguese doctoral grant, SFRH/BD/78713/2011, issued by the Fundação da Ciência e Tecnologia (FCT) in Portugal.

References

1. Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 217–253. Springer US (2011)
2. Al-Waer, H., Clements-Croome, D.J.: Key performance indicators (kpis) and priority setting in using the multi-attribute approach for assessing sustainable intelligent buildings. Building and Environment 45(4), 799–807 (2009)
3. Aztiria, A., Izaguirre, A., Augusto, J.C.: Learning patterns in ambient intelligence environments: a survey. Artif. Intell. Rev. 34, 35–51 (2010)
4. Chetty, M., Tran, D., Grinter, R.E.: Getting to green: understanding resource consumption in the home. In: Proceedings of the 10th International Conference on Ubiquitous Computing, UbiComp 2008, pp. 242–251. ACM, New York (2008)
5. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: Scenarios for Ambient Intelligence in 2010. Tech. rep., IST Advisory Group (2001), ftp://ftp.cordis.lu/pub/ist/docs/istagscenarios2010.pdf
6. Hagras, H., Doctor, F., Callaghan, V., Lopez, A.: An incremental adaptive life long learning approach for type-2 fuzzy embedded agents in ambient intelligent environments. IEEE Transactions on Fuzzy Systems 15(1), 41–55 (2007)
7. Lyon, A., Dahl: Achievements and gaps in indicators for sustainability. Ecological Indicators 17(0), 14–19 (2012), doi:10.1016/j.ecolind.2011.04.032; Indicators of environmental sustainability: From concept to applications
8. Neves, J., Ribeiro, J., Pereira, P., Alves, V., Machado, J., Abelha, A., Novais, P., Analide, C., Santos, M., Fernndez-Delgado, M.: Evolutionary intelligence in asphalt pavement modeling and quality-of-information. Progress in Artificial Intelligence 1, 119–135 (2012), doi:10.1007/s13748-011-0003-5
9. Rui, C., Yi-bin, H., Zhang-qin, H., Jian, H.: Modeling the ambient intelligence application system: Concept, software, data, and network. IEEE Trans. on SMCC 39(3), 299–314 (2009), doi:10.1109/TSMCC.2009.2014390
10. Schilit, B., Adams, N., Want, R.: Context-aware computing applications. In: First Workshop on Mobile Computing Systems and Applications, WMCSA 1994, pp. 85–90 (1994), doi:10.1109/WMCSA.1994.16
11. Singh, R., Murty, H., Gupta, S., Dikshit, A.: An overview of sustainability assessment methodologies. Ecological Indicators 9(2), 189–212 (2009)
12. Wang, K.I.K., Abdulla, W.H., Salcic, Z.: Ambient intelligence platform using multi-agent system and mobile ubiquitous hardware. Pervasive and Mobile Computing, 558–573 (2009)

Intelligent Evacuation System for Flood Disaster

Sefrioui Imane, Cherrat Loubna, Ezziyyani Mostafa, and Essaaidi Mohammed

Abstract. The main objective of this research is to find the best evacuation path during a flood disaster emergency situation. The challenge lies in how to manage this situation that has dynamic changing conditions and strict time constraints. In this paper, we propose an adaptive system for flood evacuation. This system gives the best decisions to take, in this emergency situation, to minimize damages. It computes the best evacuation routes in real time by executing an algorithm which takes into consideration the spatial characteristics of hazard propagation. The system, based on sensors network, is adapted to the changes of the environment and provides directions for the shortest and less hazardous routes to the users.

1 Introduction

Natural disasters such as floods, earthquakes, tsunamis, hurricanes, fires -in urban cities- usually result in huge environmental damages and human casualties. During these natural disasters, the emergency teams meet difficulties to design the rescue operations especially in an uncertain area which is subject to continuous changes. For example, let's assume an ambulance that wants to evacuate a victim. As the ambulance goes to rescue him, it reaches a flooded road which can't be crossed and the driver realizes that the flood blocked the evacuation path he has chosen. So, he goes back and tries to find another evacuation path hoping it won't be hazardous. However, he wastes time seeking for a safe path while the hazard has been increased.

This has shown the necessity to develop intelligent supporting evacuation systems. Thus, our research focuses on floods inundation and tries to manage them effectively taking into account the changes generated by the environment in order

Sefrioui Imane · Cherrat Loubna · Ezziyyani Mostafa · Essaaidi Mohammed
Laboratory of Information Systems and Telecommunications - Abdelmalek Essaadi University, Morocco
e-mail: {sefrioui.imane,ezziyyani,essaaidi}@gmail.com, cherrat81@yahoo.fr

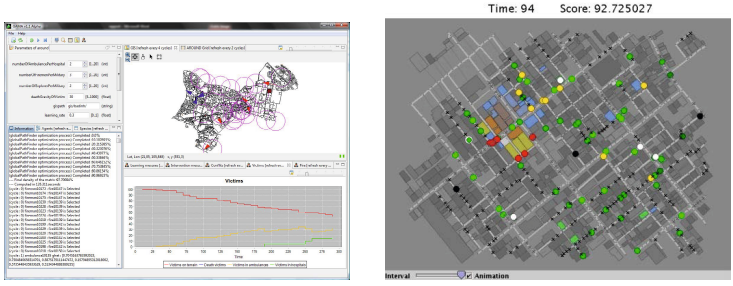
G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 205–210.
springerlink.com

to minimize the damages. So, how can we optimally plan the rescue operations in this strict time constraint, dynamic changes of the environment and the variability of factors affecting the decision making?

In this paper, we propose a simulation system that provides real-time decision support during the strict time of emergency situations in flood disaster. This system consists of collecting information, coordinating with multiple entities, allocating available resources for rescuing the victims. Our solution makes uses of a multi-agent systems and wireless sensors networks. Agents are used to represent various types of actors. The wireless sensor network is used to monitor the spread of hazards and the sensed data is used by the simulator. In addition, our system is based on outdoor navigation. So, Geographic Information Systems (GIS) are used to model the concerned areas.

2 Related Work

There is a considerable research in emergency simulation using GIS multi-agent-based models. AROUND project [1] (Autonomous Robots for Observation of Urban Networks after Disasters) is a complete simulation system for response planning in case of earthquake disasters in urban area. It is a model based on Hanoi's GIS data. It relies on a set of autonomous and communicating robots, with multiple sensors. These robots are able to self-organize in order to collect data about the damaged area. This set represents a Multi-Agents System (MAS). Each robot is viewed as an agent, able to adapt itself to face evolutions of its environment, which is unpredictable. This model is developed in GAMA platform [7] (Gis & Agent-based Modelling Architecture) which is a simulation development environment for building spatially explicit agent-based simulations (See Fig. 1a).



(a) AROUND-Rescue Model using GAMA platform.

(b) Robocup-Rescue Simulation.

Fig. 1 Simulation tools that implement the emergency response during earthquake disaster

There is another simulation tool that implements the emergency response during disasters. It is Robocup Rescue Simulation [6]. It was designed to provide emergency decision support and to simulate the rescue operations at the Hanshin-Awaji

earthquake disaster. The challenge is to rescue as many civilians and buildings as possible after an earthquake disaster (See Fig. 1b).

There are various approaches that deal with the problem of the evacuation path during emergency situations. In [2–5], the authors propose a distributed algorithm for evacuation during an emergency situation inside a building. Their system provides to evacuees the best evacuation path to the exit while a hazard is spreading inside a building. They evaluate their approach using a network of decision nodes and sensors nodes deployed in specific locations inside a building. But, their approach concerns the indoor navigation. So, we propose to extend their solution to adapt it to an outdoor environment for flood disaster.

3 Multi-Agent System (MAS)

Our multi-agent system is described as:

$$\text{System} = \{T, A, E, P\}$$

T: Task or activity that emergency centers try to solve during flood disasters such as evacuating a victim, rescuing a property, or extinguishing a fire.

A: Agents such as ambulances, fire brigades, victims, police patrols. These agents simulate human conditions and are called human agents. There are still other agents that represent sensors which explore the environment and are called hazard agents.

E: Environment where agents act such as road networks. The simulation environment is based on GIS that contain a complete and reliable spatial data.

P: Actions or protocols that agents can use to communicate with each other or interact with the environment.

4 Proposed Model

In this section, we expose the solution related to the problem described above. We begin by presenting our model. Then, we describe the algorithm proposed.

4.1 Description of the Model

We present in the following an evacuation model for flood disaster. Our work is inspired by [2–5]. We assume that our environment is represented as a graph G(V,E). This graph is based on Geographic Information System data. The Fig. 2 shows the graph representation. Vertices V are the intersection between two roads or terminal extremity of a road while edges E represent the paths that users can take.

Sensors are used to monitor the hazard and to determine the road damage degree. Each sensor is associated to a link (i,j) and measures the hazard intensity H(i,j) along this link (See Fig. 3).

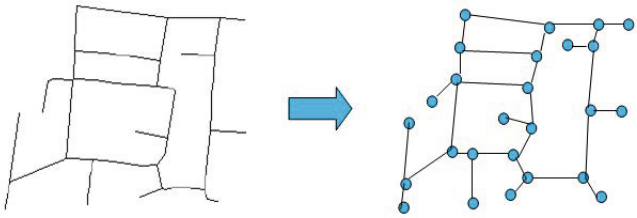


Fig. 2 GIS network and Graph representation

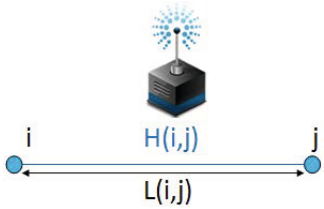


Fig. 3 Sensor that monitors the hazard intensity in a link between two nodes

In [2–5], the authors propose a model for evacuation during emergency situations inside a building. They have defined the edge cost as a function of edge length l(i,j) and hazard intensity H(i,j). The contribution of this paper lies in extending this solution for an outdoor navigation. So, we find that it is a better idea to add a new parameter T(i,j) which describes the type of users who can use this edge. Therefore, in our model, edges have multiple characteristics: edge length l(i,j) which is the distance between vertices i and j , H(i,j) which is the hazard intensity along this edge, T(i,j) which defines the possibility to use this edge by the current type of user and L(i,j) which is the cost of this edge.

Note that:

$$L(i,j) = \frac{l(i,j) \cdot H(i,j)}{T(i,j)} \tag{1}$$

H(i,j) = 1 if there is no hazard and it increases with the observed hazard. As the value of H increases, the edge becomes more hazardous to cross. If the road is completely damaged then H(i,j) = ∞ . Therefore,

$$H(i,j) \in [1, \infty[\tag{2}$$

T(i,j) = 1 if the road can be taken by the current user type and T(i,j) = 0 otherwise.

$$T(i,j) = \begin{cases} 1 \text{ if the current user type can take this edge} \\ 0 \text{ otherwise.} \end{cases} \tag{3}$$

For example, there are some roads that can be taken only by small vehicles whose height or weight doesn't exceed a certain value such as tunnels, bridges. There are also some roads for pedestrians such as sidewalk, crosswalk. Therefore, the type of user is an important parameter. If the user can take a specific road, T(i,j) = 1, then $L(i,j) = l(i,j) \cdot H(i,j)$. If he can't, T(i,j) = 0, then $L(i,j) = \infty$.

4.2 Description of the Proposed Algorithm

In this section, we propose a distributed decision (See Algorithm. 1) which is adapted to specific dynamic situations under the flood disaster. This algorithm is based on Dijkstra for the calculation of the shortest path and takes into account the hazard propagation. It is inspired by [2–5]. It is executed by each node and the output is the next node -called the Decision Node- which is on the best available path to the destination. Note that:

- **H(u, n)** is the hazard intensity along (u, n).
- **L(u, n)** is the cost of the edge (u, n).
- **C(u)** is the cost of the path from the source node s to the vertex u. Initially, this value is set to 0 for the source node (C(s) =0), and infinity for all other vertices.
- **Q** is a set of unvisited nodes. Initially, Q is a set of all the vertices.
- **argmin(Q)** defines the vertex in Q with the smallest value of C.

Initially, every node has a cost value C which is set to 0 for the initial node and infinity for the others. The Algorithm 1 marks the current node u as a visited one and removes it from the set Q of unvisited nodes. For each neighbor n of the current node u, we compute the edge cost L(u, n) and the path cost C(n). Note that L(u, n) values are updated based on measurements received from the sensor that monitors the link (u, n). Finally, this algorithm searches for the vertex u -in the set Q- that has the least cost value. This vertex is the Decision Node.

The Algorithm 1 is executed periodically until you arrive to the destination. The set of the vertices, returned by the algorithm, constitutes the evacuation path.

Algorithm 1. The decision support algorithm

Require: the node u
Ensure: the next decision node v
  Remove node u **from** the set of unvisited nodes Q
  for all neighbors, n, of u **do**
    Get H(u,n) the hazard intensity from the sensor monitoring the link (u,n)
    Compute L$(u,n) = l(i,j) \cdot H(i,j) / T(i,j)$
    C$(n) \leftarrow \min \{C(n), C(u) + L(u,n)\}$
  end for
  $v \leftarrow$ argmin $\{Q\}$
  Return v

5 Conclusion and Future Work

In this paper, we have proposed an intelligent evacuation system for a flood disaster. We have proposed a model and an algorithm for computing the shortest and safest path towards the destination to evacuate persons. This solution is based on Dijkstra algorithm and takes into account the dynamic hazards occurring in the environment.

In future, we plan to implement this model, analyze its simulation results and measure its performance during various evacuation scenarios. Also, we are studying various types of sensors to select the appropriate ones for the system.

References

1. Boucher, A., Canal, R., Chu, T.Q., Drogoul, A., Gaudou, B., Le, V.T., Moraru, V., Nguyen, N.V., Nguyen Vu, Q.A., Taillandier, P., Semp, F., Stinckwich, S.: The AROUND project: Adapting robotic disaster response to developing countries. In: IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR). Denver, Colorado (2009)
2. Dimakis, N., Filippoupolitis, A., Gelenbe, E.: Distributed building evacuation simulator for smart emergency management. The Computer Journal 53(9), 1384–1400 (2010)
3. Filippoupolitis, A., Gelenbe, E.: A distributed decision support system for building evacuation. In: Proceedings of the 2nd IEEE International Conference on Human System Interaction, Catania, Italy, pp. 323–330 (2009)
4. Filippoupolitis, A., Loukas, G., Timotheou, S., Dimakis, N., Gelenbe, E.: Emergency response systems for disaster management in buildings. In: Proceedings of the NATO Symposium on C3I for Crisis, Emergency and Consequence Management, Bucharest, Romania, pp. 1–14 (2009)
5. Gorbil, G., Filippoupolitis, A., Gelenbe., E.: Intelligent Navigation Systems for Building Evacuation. In: Computer and Information Sciences. LNEE, pp. 339–345 (2011)
6. Robocup-Rescue simulator, http://www.rescuesystem.org/robocuprescue/
7. Taillandier, P., Vo, D.A., Amouroux, E., Drogoul, A.: GAMA: a simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In: The 13th International Conference on Principles and Practices in Multi-Agent Systems (PRIMA), Kolkata, India, pp. 242–258 (2012)

Integrating Jade and MAPS for the Development of Agent-Based WSN Applications

Mariusz Mesjasz, Domenico Cimadoro, Stefano Galzarano, Maria Ganzha, Giancarlo Fortino, and Marcin Paprzycki

Abstract. Recent years have seen rapid advancements in wireless sensor networks (WSNs) and software agents resulting, among others, in maturation of their technology platforms. Furthermore, benefits of combining these research areas have been analyzed. The MAPS agent platform allows fusion of agents and WSNs. However, due to the hardware limitation, MAPS misses important functionalities needed, for instance, in advanced decision support. Such functions are available, among others, in the JADE agent platform, geared towards more powerful computing devices. Therefore, integration of MAPS and JADE had to be considered. The aim of this paper is to discuss technical issues involved in achieving this goal.

1 Introduction

Nowadays, wireless sensor networks (WSNs) and (multi-)agent systems (MAS) became popular and fast advancing research areas. Furthermore, it has been suggested that combining WSNs and MASs can bring about a new generation of intelligent systems [6, 14, 15]. One of the observable advances in both areas is the rapid maturation of their technology platforms. Currently, there exist four Java-based mobile

Mariusz Marek Mesjasz
Warsaw University of Technology, Warsaw, Poland
e-mail: mesjaszm@gmail.com

Domenico Cimadoro · Stefano Galzarano · Giancarlo Fortino
DEIS – University of Calabria, Rende (CS), Italy
e-mail: g.fortino@unical.it

Maria Ganzha
Systems Research Institute Polish Academy of Sciences and University of Gdansk, Poland
e-mail: Maria.Ganzha@ibspan.waw.pl

Marcin Paprzycki
Systems Research Institute Polish Academy of Sciences, Warsaw, Poland
e-mail: Marcin.Paprzycki@ibspan.waw.pl

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 211–220.
springerlink.com

agent platforms for WSNs: MAPS [3, 8], TinyMAPS [7], AFME [13] and MASPOT [12]. MAPS will be introduced in Section 2 whereas TinyMAPS is a compact and constrained version of MAPS ported on the Sentilla JCreate sensor platform [4]. The AFME framework, is a lightweight version of the AgentFactory framework, ported onto the Sun SPOT [5]. However, AFME was not specifically designed for WSNs and, in particular, for the Sun SPOT environment. MASPOT is a brand new mobile agent system natively designed for the Sun SPOTs. It remains to be seen if its Sourceforge code base will be maintained and updated.

Observe that the hardware capabilities of most sensor networks are restricted by the sensor miniaturization, and the battery life. Therefore, regardless of the WSN-agent platform, agents cannot be expected to perform resource-consuming tasks (e.g. extensive data analysis needed for complex decision support), which require more robust hardware and software. Therefore, in a system supporting a glider pilot (see, [10, 11]) we have proposed a hybrid approach. There, we have decided to proceed with a design of a system, in which MAPS agents are responsible for managing on-board sensors, while JADE agents ([9]) run on a smart device and are responsible for the meta-level "intelligence." For the resulting GliderAgent system we have designed and implemented an initial version of the *gateway* supporting MAPS-JADE integration (inter-communication). The *gateway* was later consolidated and enhanced. The aim of this paper is to discuss, in detail, technical issues involved in its design and implementation.

However, before we proceed, let us note that the mentioned hybrid design can be used in a large number of intelligent systems useful in many application domains such as e-health, smart homes, smart grid, or military scenarios. There, the agent-based WSN subsystem will be dealing with low-level functionalities, e.g. sensor data preprocessing, information routing, or power management. Conversely, the meta-level agent-based subsystem will facilitate data analysis, data mining, or preparation of updated strategies for agents in the WSN subsystem. Hence, the meta-level subsystem will provide the core intelligence of the system.

The remaining parts of the paper are organized as follows. Sections 2 and 3 provide brief introduction of MAPS and JADE agent platforms. The MAPS-JADE Gateway architecture is described in detail in Section 4, whereas preliminary evaluation can be found in Section 5. Finally, concluding remarks, including on-going work, complete the paper.

2 MAPS Agent Framework

MAPS [3, 8] is a Java-based framework developed for the Sun SPOT technology to enable agent-oriented programming of WSN applications. It has been conceptualized around the following requirements:

- Component-based lightweight agent server architecture to avoid heavy concurrency and agents cooperation models.
- Lightweight agent architecture to efficiently execute and migrate agents.

- Minimal core services involving agent migration, agent naming, agent communication, timing and sensor node resources access (sensors, actuators, flash memory, switches and battery).
- Plug-in-based architecture extensions, through which other services can be defined in terms of one or more dynamically installable components implemented as single or cooperating (mobile) agent/s.

The MAPS architecture (see Fig. 2) is based on components that interact through *events* and facilitate message transmission, agent creation, agent cloning, agent migration, timer handling, and access to sensor nodes.

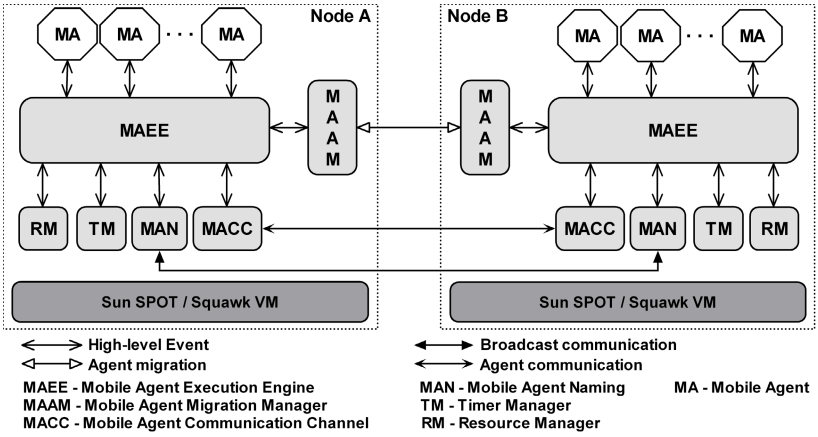


Fig. 1 MAPS architecture – an overview

In particular, the main components of MAPS are:

- *Mobile Agent (MA)* – the basic high-level component, defined by the application programmer.
- *Mobile Agent Execution Engine (MAEE)*, which manages execution of MAs using an event-based scheduler (enabling lightweight concurrency). MAEE interacts with other components, to fulfill service requests issued by MAs (e.g. message transmission, sensor reading, timer setting).
- *Mobile Agent Migration Manager (MAMM)*, which supports agents migration through the Isolate hibernation/dehibernation mechanism provided by the Sun SPOT environment [5]. While MAs hibernation and serialization involve data and execution state, the code has to reside at the destination node (this is a current limitation of the Sun SPOTs which do not support dynamic class loading and code migration).
- *Mobile Agent Communication Channel (MACC)*, enables inter-agent communications based on asynchronous messages supported by the radiogram protocol.
- *Mobile Agent Naming (MAN)*, provides agent naming based on proxies, for supporting MAMM and MACC in their operations. MAN also manages the

(dynamic) list of the neighbor sensor nodes (updated through a beaconing mechanism based on broadcast of messages).
- *Timer Manager (TM)*, supporting the timer service for timing MA operations.
- *Resource Manager (RM)*, which enables access to the resources of the Sun SPOT node: sensors, switches, leds, battery, and flash memory.

3 JADE

The Java Agent DEvelopment framework ([9]) is one of the most popular Java-based agent platforms, which complies with the Foundation for the Intelligent Physical Agents ([2]) specifications. The *JADE* agent platform is composed of a single *Main Container* and multiple *Agent Containers*, which can be distributed across different hosts. The *Main Container* contains the *Agent Management System* (*AMS*) agent and the *Directory Facilitator* (*DF*) agent. The *AMS* agent supervises the platform, manages the life-cycle of all agents inside it, and provides the *white pages* service (a registry of currently existing agents; including their *Agent Identifiers* (*AID*), used for communication). The *DF* agent provides an optional *yellow pages* service (a registry of services provided by the agents registered in the *AMS*).

All actions undertaken by agents are encapsulated within *Behaviours*, which are executed sequentially in the agent's main thread. However, if an agent has to perform a time-consuming operation (or wait for required resources), *JADE* allows a *ThreadedBehaviour*, which is executed in another thread and does not block the agent. *JADE* agents communicate via *ACLMessages*, which comply with the *FIPA ACL Message Structure Specification* [1]. In order to send an *ACLMessage*, an agent has to register an ontology and a codec. The ontology is used to demarcate data inside the *ACLMessage*. The codec encapsulates content, or extracts data from the message (based on the internal structure of the message – message header – and the ontology). The *ACLMessage* is composed of a *header* and one or more *message elements*. The message *header* is always present, and contains information necessary to properly deal with the message (e.g. the source *AID*, the target *AID*, message type, ontology, language (codec), performative, etc.). In *JADE*, message elements can be of a primitive type (e.g., *boolean*, *int*, *string*), or of an aggregate type (any user-defined structure composed of primitive or aggregate elements). Aggregate elements are usually represented as Java classes included in the ontology. Message elements form a content of the *ACLMessage*, which can is represented depending on the codec. For example, a codec can write data in a human readable form (XML, strings), or as a byte code.

4 Gateway Architecture

As stated above, the *JADE/MAPS gateway* (or, simply, the *gateway*) has been implemented to provide a communication mechanism between JADE and MAPS agents. It facilitates bi-directional translation between *JADE ACL messages* and *MAPS Events* and supports routing of communication between the two agent platforms.

The *gateway* is composed of two abstract parts – the *JADE part* and the *MAPS part*, responsible for communication with their own platforms. These two parts interact within the *gateway* using *translation mechanisms*. The *JADE part* of the *gateway* is a *JADE agent*. It was a natural choice, because JADE agents can autonomously perform complex tasks. Furthermore, it was established that the translation and routing mechanism should be accomplished with the more powerful environment, thus resulting in assigning this role to a JADE agent that runs on a PC (as the primary target environment).

For the *MAPS part*, there was no simple way to connect the *gateway* to the MAPS platform. Placing any part of the *gateway* on a Sun SPOT device could result in a communication bottleneck due to its limited resources. Since it is impossible to run MAPS agents without the MAPS Execution Engine, to allow its execution outside of the Sun SPOT devices (e.g. on a PC), the MAPS platform would have to be rewritten. However, implementation of a fully functional PC-based MAPS platform would not help solving the cross-platform communication problem. Instead, it was enough to reimplement selected parts of the MAPS Execution Engine that (i) are responsible for communication over the radio, and (ii) manage the list of remote MAPS agents. Hence, the *gateway* was developed as a *semi-functional MAPS Execution Engine* without the capability of running actual MAPS agents.

Let us now describe the *JADE-MAPS communication* from three perspectives: (1) communication within JADE, (2) communication within MAPS, and (3) passing information between the two platforms. Recall that the *gateway* is a *JADE agent*. Therefore, it can be directly accessed by other JADE agents via the *AMS* and *DF* services (see section 3), and can communicate with them using ACL messages. However, to facilitate JADE-MAPS communication, the *gateway* provides a special ontology (*GatewayOntology*), which describes actions that need to be performed by the *gateway*: *Register*, *Unregister*, *GetRemoteAgent* and *SendMessage*. Each action is represented by a Java class, which is used to fill the content of an *ACLMessage*. The ACL message containing one of these actions has the performative set to *REQUEST*. The *gateway* can respond to such messages with an *INFORM* message, containing a confirmation of execution of the requested action (and a list of MAPS agents, in the case of *GetRemoteAgents*), or with a *FAILURE* message containing a string, specifying the cause of the failure (see Fig. 2). If the JADE agent requests communication by sending the *SendMessage* action, the *gateway* may optionally send an additional *INFORM* message with a response from a MAPS agent (an instance of the *ReceivedMessage* class, also included in the *GatewayOntology*).

A JADE agent, which wants to communicate with MAPS agents, has to register itself within the *gateway*. To do this, the agent has to send an ACL message specifying the *Register* action. When the *gateway* receives such a message, it creates a unique *MAPS ID* (for the sender's *AID*) and sends it within a *MAPS REFRESH* message to the *MAPS Execution Engines*, to allow agent synchronization across the platform. Next, the *gateway* adds a pair (*AID*, *MAPS ID*) to its list of local agents. These agent identifiers are used during the message translation process. All registered *JADE agents* should unregister themselves at the end of their life-cycle by sending an ACL message with the *Unregister* action. In response, the *gateway*

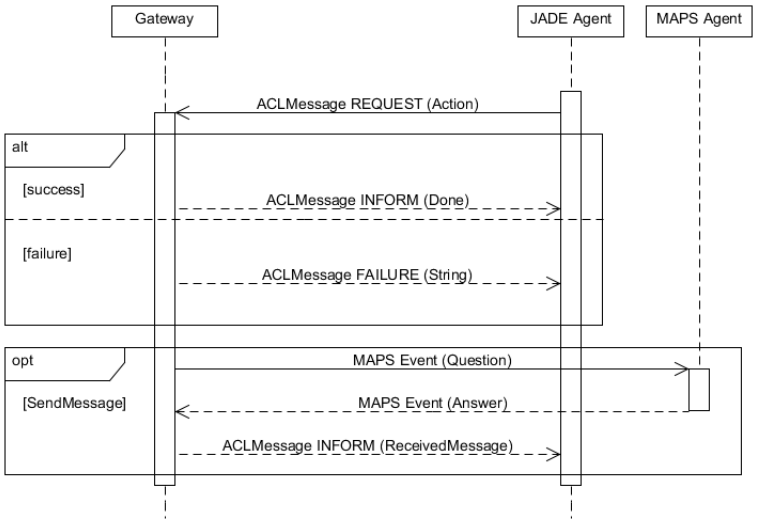


Fig. 2 A sequence diagram which represents a typical communication with the *gateway*

removes the pair (*AID*, *MAPS ID*) from the memory and sends a *MAPS REFRESH* message to the *MAPS Execution Engines*, informing that a given *MAPS ID* is no longer valid.

From the MAPS perspective, the *gateway* is just another *MAPS Execution Engine*. To facilitate this, the *gateway* broadcasts a *MAPS PUBLISH* message, which (1) makes the *gateway* available within the MAPS platform, (2) discovers all available Sun SPOT devices that run MAPS, and (3) initializes the lists of MAPS agents. This list is later updated, based on the communication between the *gateway* and the other MAPS Execution Engines.

The JADE agents are accessible to the MAPS agents by their *MAPS IDs*, which correspond to their *AIDs*. The MAPS agents are *not* aware that the *gateway* (and the JADE agents) belong to a different platform. Hence, communication between the MAPS and the JADE agents is simplified to communication between MAPS agents.

The translation mechanism "combines" the two platforms and is invoked when the *gateway* receives a message, which has to be translated either into an ACL message, or into a MAPS Event. The translation begins with the creation of an appropriate message header. Since an ACL message and a MAPS Event are very different, the *gateway* had to introduce a common communication standard. As presented in Fig. 3, the ACL message contains the message header and a *SendMessage* action, which not only requests to send a message, but also provides additional information encapsulated within the *Message* concept. The *Message* concept includes a source *AID*, a target *MAPS ID*, a message name (here, topic 41 means that this is an ordinary message), a message type (occurrence time 1, means "now") and parameters. Obviously, this information is MAPS-specific and does not have an equivalent in the ACL message (for example, the message name *TEMPERATURE* does not match

```
ACLMessage - question :

(REQUEST
 :receiver  (set (agent-identifier :name Gateway@192.168.1.3:1099/JADE
                    :addresses (sequence http://Mariusz-Laptop:7778/acc)) )
 :content  "((action (agent-identifier :name Gateway)
    (Send
        (Message
            :agentAID (agent-identifier
            :name Peter@192.168.1.3:1099/JADE
                :addresses (sequence http://Mariusz-Laptop:7778/acc))
            :mapsID \"0000.12345ABCD\"
            :msgName 41
            :msgType 1
            :parameters (sequence question \"how are you\"))))"
 :language  fipa-sl  :ontology  JADE-MAPS-gateway-ontology )

ACLMessage - question (translation to MASP Event) :

0000.9876ZYXW$0000.12345ABCD$41$1$question&how are you&

MAPS Event - reponse :

0000.12345ABCD$0000.9876ZYXW$41$1$answer&thanks im fine&

MAPS Event - reponse (translation to ACLMessage) :

(INFORM
 :receiver  (set (agent-identifier :name Peter@192.168.1.3:1099/JADE
                    :addresses (sequence http://Mariusz-Laptop:7778/acc)) )
 :content  "(
    (Message-for-you
        (Message
            :agentAID (agent-identifier :name Peter@192.168.1.3:1099/JADE
                :addresses (sequence http://Mariusz-Laptop:7778/acc))
            :mapsID \"0000.12345ABCD\"
            :msgName 41
            :msgType 1
            :parameters (sequence answer \"thanks im fine\"))))"
 :language  fipa-sl  :ontology  JADE-MAPS-gateway-ontology )
```

Fig. 3 An example of the simple translation mechanism. The *ACLMessages* has been encoded by the *SLCodec*.

any property in the ACL message). A corresponding *MAPS Event* header is created, based on this information. The only field that requires translation is the agent *AID* (valid only inside the JADE platform). Here, the *gateway* translates the *AID* based on the list of (*AID*, *MAPS ID*) pairs (e.g. in Fig. 3, the *gateway* found the pair (*Peter@192.168.1.3:1099/JADE*, *0000.98765XYZW*) and used it in the corresponding MAPS Event).

The translation from a MAPS Event to an ACL message is the reverse process. The *gateway* starts with an empty *ACLMessage* class of the type *INFORM*. The *AID*, corresponding to the target *MAPS ID* (from the MAPS Event), is set as the receiver of this message. However, the *gateway* has to specify the context of the MAPS Event (the message name and the message type). Since the ACL message header does not contain the MAPS-specific information, the *GatewayOntology* provides a predicate called *ReceivedMessage* (denoted by the string "Message-for-you"). The predicate contains only one field, which stores the *Message* concept. This *Message* concept is filled with MAPS information by the *gateway*. Notice that the ACL messages, presented in Fig. 3, differ from each other to a small extent. Namely, the ACL message type changes from *REQUEST* to *INFORM* and, respectively, the content of the

message is changed from the *SendMessage* action to the *ReceivedMessage* predicate. Otherwise, the translation mechanism does not alter the internal structure of MAPS Events.

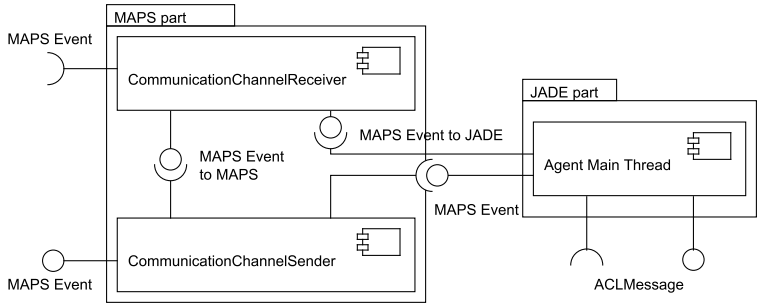


Fig. 4 A component diagram of the gateway

Technically, the *gateway* uses three threads: (i) *CommunicationChannelReceiver* (*CCR*), (ii) *CommunicationChannelSender* (*CCS*), and (iii) the *Agent Main Thread* (see Fig. 4). The *CCR* and the *CCS* originate from the *MAPS Execution Engine* source code, and are used inside the *gateway* as separate *ThreadedBehaviour*s. The *CCR* has to be run as a separate thread due to the asynchronous communication within the *MAPS platform*. The execution of the CCS as a different thread guarantees the absence of communication bottleneck, as discovered during tests on an earlier version of the gateway (see [10]).

Each time the *CCS* receives a MAPS Event, the datagram must be checked for the destination address. If the MAPS Event is addressed to another MAPS agent or an Execution Engine within the MAPS platform, it is forwarded directly. However, if the MAPS Event has to be delivered to a JADE agent (with a *MAPS ID* available in the *gateway*), then it is sent to the *Agent Main Thread* for further processing.

For each message, the agent main thread has to determine its destination. If this is a local JADE-to-JADE message, it is processed inside the *JADE part* of the *gateway*. If the message has to be sent to a MAPS agent, then the translation mechanism is invoked and the corresponding *MAPS Event* is put in the *CCR* queue, to be forwarded to an appropriate MAPS Execution Engine. Note that the *gateway* introduces two restrictions: (a) two JADE agents inside the same gateway cannot communicate with each other via MAPS Events, and (b) there is no confirmation that a MAPS agent will receive, or respond to, an ACL message. The first restriction is to reduce the workload of the *gateway*. The second restriction is due to lack of a mechanism to monitor message traffic in the radio network used by the MAPS platform.

5 Preliminary Performance Test

To check the performance of the *gateway*, we executed a communication test between a number of agent pairs. Each pair was composed of one JADE and one MAPS agent. This test was designed to flood the system with ACL messages and MAPS Events. During the test, we incremented the number of agent pairs from 1 to 20. Frequency of message transmission was set to 500 milliseconds, and the message size was 8 bytes (representing the standard size of the value of temperature) plus the size of the message header. The *Forwarding Time* (*FT*) was measured, and represents the time needed by the gateway to forward a message from the Jade agent, plus the time needed to receive a reply form the MAPS agent ("round-trip" communication). Specifically, during each experiment, a sender (*JADE agent*) sends a message to a receiver (*MAPS agent*) to request the value of the temperature, and receives a response. Each test lasted 5 minutes to complete, and a total of 20 experiments have been carried out to obtain a good confidence measure.

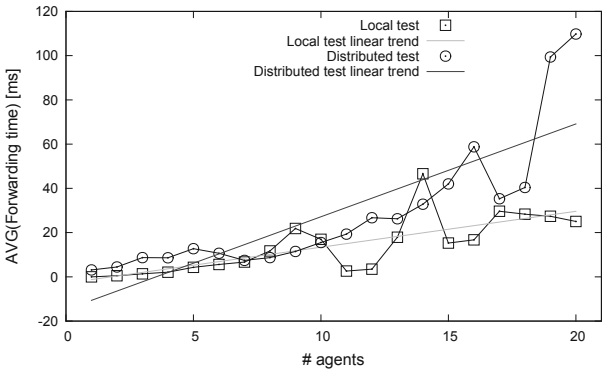


Fig. 5 Forwarding time for multiple agents

The experimental results are presented in Fig. 5. The *FTs* for the distributed test (agents on different machines) are greater than the ones obtained on a single host (due to the use of the RMI protocol to communicate between JADE agents and the *gateway*). This is the reason why the linear trend corresponding to the local test grows faster and its slope is steeper. Moreover, the increase in the number of agents results in a greater variation of the *FT*. Evaluation shows also the performance degradation due to the time-consuming operations (serialization and radio stream-based communication) performed by the *Sun SPOT libraries*. Hence, the performance of the *gateway* is highly influenced by the Sun SPOT emulator (the *Solarium*).

6 Concluding Remarks

In this paper we have proposed and described a *gateway* component providing inter-communication capabilities between MAPS and JADE agent platforms. The

importance of such gateway is in providing better support for the development of intelligent WSN systems, where the small footprint MAPS agents facilitate sensor management, whereas JADE agents infuse the system with intelligence. In the near future, we plan to carry out more complete performance evaluations of the *gateway*, and to include it in the JADE plug-in repository.

References

1. Fipa acl message structure specification, http://www.fipa.org/specs/fipa00061/SC00061G.html
2. The foundation of intelligent physical agents (FIPA) (March 13, 2010), http://www.fipa.org/
3. Mobile Agent Platform for Sun SPOT (MAPS) (2011), http://maps.deis.unical.it
4. Sentilla developer community (2011), http://www.sentilla.com/developer.html
5. Sun Small Programmable Object Technology (Sun SPOT), documentation and software, http://www.sunspotworld.com (2011)
6. Aiello, F., Fortino, G., Galzarano, S., Gravina, R., Guerrieri, A.: An analysis of java-based mobile agent platforms for wireless sensor networks. Multiagent and Grid Systems 7(6), 243–267 (2011)
7. Aiello, F., Fortino, G., Galzarano, S., Vittorioso, A.: TinyMAPS: A Lightweight Java-Based Mobile Agent System for Wireless Sensor Networks. In: Brazier, F.M.T., Nieuwenhuis, K., Pavlin, G., Warnier, M., Badica, C. (eds.) Intelligent Distributed Computing V. SCI, vol. 382, pp. 161–170. Springer, Heidelberg (2011)
8. Aiello, F., Fortino, G., Gravina, R., Guerrieri, A.: A java-based agent platform for programming wireless sensor networks. The Computer Journal 54(3), 439–454 (2011)
9. Bellifemine, F., Poggi, A., Rimassa, G.: Developing multi-agent systems with a fipa-compliant agent framework. Softw., Pract. Exper. 31(2), 103–128 (2001)
10. Domanski, J.J., Dziadkiewicz, R., Ganzha, M., Gab, A., Mesjasz, M.M., Paprzycki, M.: Implementing glideragent-an agent-based decision support system for glider pilots. In: Software Agents, Agent Systems and Their Applications, pp. 222–244 (2012)
11. Gab, A., Adreout, P., Ganzha, M., Paprzycki, M.: Glideragent-a proposal for an agent-based glider pilot support system. In: Proceedings of the 15th International Conference on Methods and Models in Automation and Robotics (MMAR), pp. 55–60. IEEE Press (2010)
12. Lopes, R., Assis, F., Montez, C.: MASPOT: A Mobile Agent System for Sun SPOT. In: Proceedings of the 2011 Tenth International Symposium on Autonomous Decentralized Systems, ISADS 2011, pp. 25–31. IEEE Computer Society, Washington, DC (2011)
13. Muldoon, C., O'Hare, G., O'Grady, M., Tynan, R.: Agent migration and communication in WSNs. In: 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 425–430. IEEE (2008)
14. Rogers, A., Corkill, D.D., Jennings, N.R.: Agent technologies for sensor networks. IEEE Intelligent Systems 24, 13–17 (2009)
15. Vinyals, M., Rodríguez-Aguilar, J.A., Cerquides, J.: A survey on sensor networks from a multi-agent perspective. The Computer Journal 54(3), 455–470 (2010)

A Collaborative Bayesian Watchdog for Detecting Black Holes in MANETs

Manuel D. Serrat-Olmos, Enrique Hernández-Orallo, Juan-Carlos Cano, Carlos T. Calafate, and Pietro Manzoni

Abstract. Watchdogs are a well-known mechanism to detect threats and attacks from misbehaved and selfish nodes in computer networks. In infrastructureless networks, such as MANETs, attack detection and reaction is a key issue to the whole network. Watchdog systems overhear traffic and perform analysis using data collected to decide the grade of misbehaviour that their neighbour nodes present, so accuracy and detection speed play a key role in achieving the right level of network security and performance. The problem behind the use of watchdogs is that they can cause a relatively high level of false positives and false negatives, so increasing their accuracy is also basic to successfully implement this technology in real MANET environments. This paper proposes a collaborative approach for detecting black holes and selfish nodes in MANETs, using a set of watchdogs which collaborate to enhance their individual and collective performance. The paper shows that using this collaborative watchdog approach the detection time of misbehaved nodes is reduced and the overall accuracy increased.

1 Introduction

A Mobile Ad Hoc Network, usually known as MANET, consists in a set of wireless mobile nodes that function as a network in the absence of any kind of centralized administration and networking infrastructure. These networks rely on cooperation from their nodes to correctly work, that is, every network node generates and sends its own packets and forwards packets in behalf of other nodes. These nodes could be classified [9] as well-behaved nodes, if they cooperate with the MANET forwarding

Manuel D. Serrat-Olmos · Enrique Hernández-Orallo · Juan-Carlos Cano · Carlos T. Calafate · Pietro Manzoni
Departamento de Informática de Sistemas y Computadores
Universidad Politécnica de Valencia. Valencia, Spain
e-mail: mdserrat@upvnet.upv.es, {ehernandez,jucano,calafate,pmanzoni}@disca.upv.es

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 221–230.
springerlink.com

activities to achieve the community goals, or as misbehaved nodes, if they act against those global goals. In this case, nodes are further classified into three classes: faulty nodes, if they do not cooperate due to a hardware or software malfunction; selfish nodes, if they drop all the packets whose destination node are not themselves, but they use other nodes to send their own packets; and malicious nodes, when they try to disturb the normal network behaviour for their own profit.

When a MANET is deployed, we have to assume that there could be a percentage of misbehaved nodes. The types of misbehaved nodes, their number, and their positions and movement patterns are key issues which deeply impact the mobile ad hoc network performance [8]. Additionally, network performance could be drastically reduced if nothing is done to cope with these threats. To this end, an effective protection against misbehaved nodes will be mandatory to preserve the correct functionality of the MANET [6].

In a MANET there are basically two kinds of packet flows: data packets and route maintenance packets. However, not all misbehaved nodes have the same impact on network performance, due to the type of packet flows they affect. A really malicious node could damage the network, spoofing routes, flooding the wireless channel, or carrying out a man-in-the-middle attack. These are classical attacks that every network could suffer, and solutions have been devised for them.

All types of misbehaved nodes –faulty, selfish and malicious– have a common behaviour: they do not participate in forwarding activities, thus being characterized as black holes. A *black hole attack* is a type of attack in which a node intends to disrupt the communication with its neighborhood by attracting all traffic flows in the network and then dropping all packets received without forwarding them to their final destination [5]. To avoid or significantly reduce this type of attack in MANETs, several proposed approaches are based on monitoring the traffic heard by every node to detect misbehaved nodes, and then taking the appropriate actions to avoid the negative effects of that misbehaviour [10].

The main problem that arises at this point is how to detect these black holes avoiding as much as possible wrong diagnostics, like false positives or false negatives. A false positive appears when the selected technique identifies a well-behaved node as a misbehaved node. A false negative appears when the technique can not detect a misbehaved node, so the network believes that it is a normal node, with its potentially disruptive effects. So, accuracy and detection speed are critical issues when designing an approach for black holes detection in MANETs.

Several solutions have been proposed for detecting and isolating or incentivating misbehaved nodes in MANETs. Marti et al. [7] proposed a Watchdog and a Pathrater over DSR protocol to detect non-forwarding nodes, maintaining a rating for every node and selecting routes with the highest average node rating. The response module of this technique only relieve misbehaved nodes from forwarding packets, but they continue getting their traffic forwarded across the network. Buchegger and Le Boudec [1] proposed the CONFIDANT protocol over DSR, which combines a watchdog, a reputation system, Bayesian filters and information obtained from a node and its neighbours to accurately detect misbehaved nodes. The system's response is to isolate those nodes from the network, punishing them indefinitely.

Others approaches do not use reputation systems, in favor of incentivation. Buttyan and Hubaux [2, 3] presented a method using a virtual currency called *nuglet*. Every node has a credit counter which will be increased when the node forwards packets, and decreased when a node sends his own packets. When a node has no nuglets, it can't send its packets, so it is a motivation for nodes to forward packets for the network benefit. Zhong et al. [11] proposed SPRITE, a credit-based system to incentivate participation of selfish nodes in MANET communication. It's based on a Central Clearance System, which charges or gives credit to nodes when they send or forward a message. So, if a node wants to send a message, it must have sufficient credit to do it. That credit is earned by forwarding messages for other nodes. The response module of this method is integrated into the incentivation method, so that if a node doesn't forward other nodes' messages, it wont have credit to send its own messages.

Many of these approaches use the concept of reputation to improve the detection of black holes, just as reputation is used in human relations. If a node group says that other node is malicious, it is quite probable that this is true. So, it seems a good idea to integrate reputation systems in the mechanism to detect misbehaved nodes. Therefore, watchdog cooperation will probably increase accuracy and detection speed.

In this work we propose a collaborative watchdog which integrates techniques from reputation systems and bayesian filtering, and makes extensive use of the collaborative nature of MANETs. This watchdog must be considered as an Intrusion Detection Systems (IDS), which is a software piece that collects and analyzes network traffic to detect a set of attacks. In this context, intrusion detection systems aim at monitoring the activity of the nodes in the network in order to detect misbehaviour [5]. Usually, this kind of software products are built using two building blocks: a Detection (or sensor) module, like watchdogs, and a Response module.

The rest of this paper is organized as follows. Section 2 presents the concept of bayesian watchdog, which is a basic technique to detect black holes in MANETs. Section 3 presents an enhanced proposal for a collaborative watchdog designed to perform that task. Section 4 evaluates its benefits and, finally, Section 4 provides some concluding remarks.

2 Bayesian Watchdog

As we stated earlier, to detect misbehaved nodes, network monitoring is needed. Every node must be aware of its neighbours' behaviour, and watchdogs are a popular component for Intrusion Detection System dedicated to this task. The main problem is that watchdogs are characterized by a significative amount of false positives [5], basically due to mobility and signal noise. Previous works from our group [4] have evaluated a bayesian watchdog over Ad-hoc On-demand Distance Vector (AODV) routing in MANETs. This bayesian watchdog results from the aggregation of a bayesian filter with a standard watchdog implementation.

The standard watchdog simply overhears the packets transmitted and received by its neighbours, counting the packets that should be retransmitted, and computing a trust level for every neighbour as the ratio of "packets retransmitted" to "packets that should have been retransmitted". If a node retransmits all the packets that it should had retransmitted, it has a trust level of 1. If a node has a trust level lower than the configured tolerance threshold, that node is marked as malicious.

The role of the bayesian filter in the watchdog is to probabilistically estimate a system's state from noisy observations [4]. The mathematical foundation of the bayesian filter is the following: at time t, the state is estimated by a random variable ϑ, which is unknown, and this uncertainty is modeled by assuming that ϑ itself is drawn according to a distribution that is updated as new observations become available. It is commonly called *belief* or $Bel_t(\vartheta)$. To illustrate this, let's assume that there is a sequence of time-indexed observations $z_1, z_2, ..., z_n, ..., z_t$. The $Bel_i(\vartheta)$ is then defined by the posterior density over the random variable ϑ conditioned on all sensor data available at time t:

$$Bel_t(\vartheta) = p(\vartheta | z_1, z_2, ..., z_n, ..., z_t) = Beta(\alpha_t, \beta_t, \vartheta) \tag{1}$$

In this approach, the random variable ϑ belongs to the interval [0,1]. Bayesian filtering relies on the Beta distribution, which is suitable to estimate the belief in this interval, as shown in expression 1; α and β represent the state of the system, and they are updated according to the following equations:

$$\alpha_{t+1} = \alpha_t + z_t \quad \beta_{t+1} = \beta_t + z_t \tag{2}$$

The Beta function only requires two parameters that are continuously updated as observations are made or reported. In this approach, the observation z_t represents the information from the local watchdog obtained in time interval $[t, t + \Delta t]$ about the percentage of non-forwarded packets. The bayesian watchdog uses three parameters: the first two parameters are α and β, which are handled over to the Beta function to obtain an estimation of the node's maliciousness. Thus, we can say that α and β are the numeric representation of a node's reputation. The third parameter is γ, which represents the devaluation that old observations must suffer to adapt the watchdog's behaviour to a continuously changing scenario without penalizing certain nodes forever. It is a mechanism to reintegrate nodes into the MANET if they change their behaviour to a more cooperative one.

As a result of their work, Hortelano et al. [6] found that, compared to the standard one, the bayesian watchdog reached a 20% accuracy gain, and it presents a faster detection on 95% of times.

3 Collaborative Bayesian Watchdog

Based on the bayesian watchdog presented in Section III, we have implemented a collaborative bayesian watchdog based on a message-passing mechanism in every individual watchdog that allows publishing both self and neighbour reputations.

Every node running our watchdog collects the reputation information to obtain the values of α' and β' for every neighbour. The underlying idea of our approach is that if a bayesian watchdog works well for detecting black holes, a group of collaborating neighbouring bayesian watchdogs would be able to perform faster and more accurate detections.

Similarly to the bayesian watchdog, the collaborative bayesian watchdog overhears the network to collect information about the packets that its neighbours send and receive. Additionally, it obtains the α and β values for its whole neighbourhood. These values are exactly the same that those obtained by the bayesian watchdog with the same observations; we call them 'first hand information' or 'direct reputations'. Periodically, the watchdog shares these data with its neighbours, and we call them 'second hand information' or 'indirect reputation'. In our implementation, indirect reputations are modulated using a parameter δ. Whenever requiered, every node running the collaborative bayesian watchdog calculates, using expressions (4) and (5), the values of α' and β', which in this case are passed to the beta function to obtain an estimation of the maliciousness of a node.

$$\underset{j\in N_i}{\forall}\ \underset{k\in N_j}{\forall}\ \alpha(i)'_j = \frac{\alpha(i)_j + \delta mean(\alpha(i)^k{}_j)}{2} \tag{3}$$

$$\underset{j\in N_i}{\forall}\ \underset{k\in N_j}{\forall}\ \beta(i)'_j = \frac{\beta(i)_j + \delta mean(\beta(i)^k{}_j)}{2} \tag{4}$$

where

- i is the node which is performing detection
- N_i is the neighbourhood of node i
- $\alpha(i)_j$ is the value of α calculated for every neighbour j of i, obtained from direct observations at i
- $\beta(i)_j$ is the value of β calculated for every neighbour j of i, obtained from direct observations at i
- $\alpha(i)^k{}_j$ is the value of α calculated for every neighbour j of i, obtained from observations of every neighbour k *of* j
- $\beta(i)^k{}_j$ is the value of β calculated for every neighbour j of i, obtained from observations of every neighbours k *of* j
- δ represents the level of trust or the relative importance that a neighbour's observed reputations have for node i

When indirect reputations arrive at a node from one of its neighbours, it only processes those reputations for its own neighbours, because reputations about nodes that are not in its neighbourhood are useless. Once the reputations have been obtained, and the adequate analysis have been done, the detection only needs a predefined tolerance threshold to identify if a node is misbehaved or not.

Figure 1 shows the main components of the collaborative bayesian watchdog. First, each individual watchdog overhears the network to make direct observations of its neighbours, thereby detecting black holes as the bayesian watchdog does.

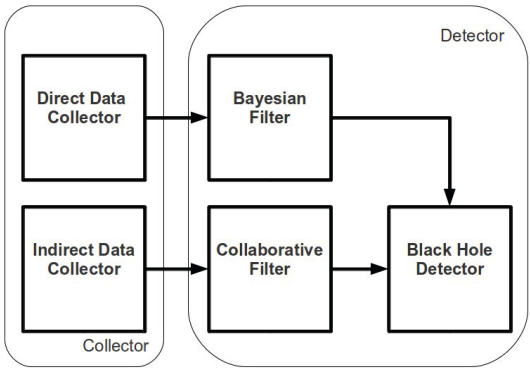


Fig. 1 Main components of the collaborative bayesian watchdog

Periodically, it receives reputation information from its neighbours and evaluates their behaviour taking into account this second hand information and its direct observations.

The algorithm of the detector module is outlined in Algorithm 1. Basically, the BayesianDetection function performs analysis over direct observations, obtaining the values of α and β. The relationship between α and β exceeds a predefined tolerance level, the watchdog identifies that node as malicious. Thes values of α and β are also used in the CollaborativeDetection function, according to expressions (4) and (5). This function operates in a similar way that the BayesianDetection function, although it uses second hand information and other parameters to perform its task.

Having introduced both the mathematical model and the algorithms designed, we now set the objectives we are trying to achieve with this collaborative bayesian watchdog. In this case, we want to:

1. increase the detection speed, reducing the time needed to detect a black hole
2. reduce the production of false positives
3. reduce the production of false negatives

4 Evaluation

We have implemented our collaborative bayesian watchdog as a Network Simulator 2 (ns-2) extension to the AODV routing protocol. We evaluate the impact that our approach has over the accuracy and the detection speed. We compare the results from the collaborative bayesian watchdog with those obtained using the non-collaborative versions, both bayesian and standard. Table 1 shows the characteristics of the scenarios we have selected for our performance evaluation.

Some of these parameters, like area, number of nodes or speed, are needed by ns-2 to execute the simulation. Others, like δ, γ, or *Observation time*, are needed by our code to perform its functionality. For each test, we averaged the results of 20 independent simulations. To obtain normalized results, we simultaneously executed

Algorithm 1. Black Hole Detector processing algorithm

```
Every observation_time Do
  For all Node_j which is a neighbour
    If ( BayesianDetection() or CollaborativeDetection() )
    Then Node_j is malicious
    EndIf
  EndFor
EndEvery
Function BayesianDetection()
  Obtain observations
  Compute α and β
  If relationship between α and β exceeds tolerance
  Then return true
  Else return false
  EndIf
EndFunction
Function CollaborativeDetection()
  Obtain neighbourhood reputations
  Compute α' and β'
  If relationship between α' and β' exceeds tolerance
  Then return true
  Else return false
  EndIf
EndFunction
```

a simulation of the standard watchdog, the bayesian watchdog, and the collaborative bayesian watchdog with the same scenarios and parameters.

4.1 Detection Speed

Accuracy is a key issue when detecting black holes, but speed is also important. A watchdog that detects 100% of black holes but requires 10 minutes is a useless watchdog. So, it is crucial for accuracy and speed to be well balanced. In that sense, watchdog enhancements will target both speed and accuracy issues.

The collaborative bayesian watchdog performed well in terms of speed. Table 2 shows that, on average, 7% of the times our approach detected black holes before the bayesian watchdog, with the same traffic pattern. The rest of the cases, it detects the malicious nodes at the same time. When a node B enters[1] node A's neighbourhood, our approach allows node A to identify node B as a black hole with only a reputations sharing phase with its common neighbours. This means that even if node B does not send or receive any data or routing packet when enters node A's neighbourhood, if it has been previously detected as black hole, node A will quickly mark it as a black hole too.

[1] In this context, entering a node's neighbourhood means that this node is in communication range and it announces its presence, for example, with a standard HELLO message.

Table 1 Simulation parameters

Parameter	Value
Nodes	50
Area	1000 x 1000 m.
Wireless interface and bandwith	802.11 at 54 Mbps
Antenna	Onmidirectional
Node speed	5, 10, 15 and 20 m/s.
% of black holes	10%
δ	0.8
γ	0.85
Fading	1
Neighbour time	1s.
Observation time	0.2s.
UDP Unicast traffic	Three flows
UDP Broadcast traffic	every 5s.
Simulation time	352 s.
Scenarios	20

Table 2 Percentage of detections where the Collaborative Bayesian Watchdog detects the black holes before the Bayesian Watchdog does it

Node Speed (m/s.)	Percentage of earlier detections
5	1.04%
10	11.88%
15	9.66%
20	5.72%

In dense networks with traffic load equally balanced between malicious and well-behaved nodes, both watchdog versions will perform nearly equally, despite of the smaller number of packets that the collaborative bayesian watchdog needs to perform detections. This is because the interval between packets is very short. Nevertheless, in networks with low traffic load and with black holes that transmit a very small amount of packets, the difference of performance between the two approaches could be more significative in terms of time. A single packet would make the difference between detecting or not a black hole, and the collaborative bayesian watchdog obtains better results in those cases.

Additionally, we can say that the collaborative bayesian watchdog obtains the best results at node speed of 10 m/s. In fact, when node move at 10 m/s and 20 m/s our approach behaves nearly 12% and 6% better respectively. These results lead to the conclusion that the collaborative bayesian watchdog becomes a suitable implementation for Vehicle Area Networks, or VANETs, a type of MANET formed by vehicles in movement which share data when they cross with another car, or communicate with a fixed network infrastructure.

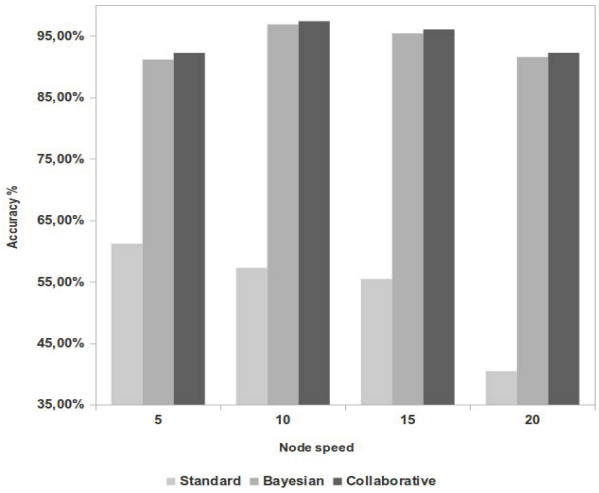


Fig. 2 Accuracy comparison of the different watchdog versions

4.2 Accuracy

Figure 2 shows that the accuracy in detecting false positives and false negatives is also slightly better than with the non-collaborative bayesian watchdog, which comes from the decreased level of false negatives. The fact is that a small amount of black holes, that are not detected with the bayesian watchdog, are now detected by the collaborative bayesian watchdog. In fact, our approach is able to detect cases where a black hole enters and exits from the range of a watchdog quickly. As shown in Figure 2, although there is not a big difference between them, the collaborative bayesian watchdog performs better in terms of accuracy than the bayesian watchdog, despite of the node speed[2]. With respect to the standard watchdog, our approach clearly surpasses it in terms of detection accuracy.

5 Conclusions and Future Work

In this paper we showed that a bayesian watchdog performs better than a standard watchdog, reducing the amount of false positives. We have also demonstrated that analyzing second-hand information using a collaborative bayesian watchdog will also help at boosting its performance by decreasing the amount of false negatives and speeding up the detection process. As a result, in the scenarios we tested our approach improves detection speed of black holes, and slightly increases the accuracy of that detection process. These conclusions evidence that, compared to previous

[2] The standard watchdog has a poor performance, as stated in [6] and as shown in Figure 2. Further comparisons with that version of the watchdog will not be done.

versions, our watchdog technique fits not only generic MANET environments, but also VANET environments.

As future work, we will implement this mechanism in a hardware testbed (Castadiva), working on the fine tuning of the collaborative bayesian watchdog.

Acknowledgements. This work was partially supported by the *Ministerio de Ciencia e Innovación*, Spain, under Grant TIN2011-27543-C03-01.

References

1. Buchegger, S., Le Boudec, J.Y.: Self-policing mobile ad hoc networks by reputation systems. IEEE Communications Magazine (2005)
2. Buttyan, L., Hubaux, J.P.: Enforcing service availability in mobile ad-hoc wans. In: IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing, MobiHOC (2000)
3. Buttyan, L., Hubaux, J.P.: Stimulating cooperation in self-organizing mobile ad hoc networks. Mobile Networks and Applications 8(5) (2003)
4. Hortelano, J., Calafate, C.T., Cano, J.C., de Leoni, M., Manzoni, P., Mecella, M.: Blackhole attacks in p2p mobile networks discovered through bayesian filters. In: Proceedings of OTM Workshops 2010, pp. 543–552 (2010)
5. Hortelano, J., Cano, J.C., Calafate, C.T., Manzoni, P.: Watchdog intrusion detection systems: Are they feasible in manets? In: XXI Jornadas de Paralelismo, CEDI 2010 (2010)
6. Kargl, F., Klenk, A., Schlott, S., Weber, M.: Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks. In: Castelluccia, C., Hartenstein, H., Paar, C., Westhoff, D. (eds.) ESAS 2004. LNCS, vol. 3313, pp. 152–165. Springer, Heidelberg (2005)
7. Marti, S., Giuli, T., Lai, K., Baker, M.: Mitigating routing misbehavior in mobile ad hoc networks. In: Proceedings of the Sixth International Conference on Mobile Computing and Networking, MobiCom 2000 (2000)
8. Sundarajan, T., Shammugam, A.: Modeling the behavior of selfish forwarding nodes to stimulate cooperation in manet. International Journal of Network Security and its Applications (IJNSA) 2(2) (2010)
9. Toh, C.K., Kim, D., Oh, S., Yoo, H.: The controversy of selfish nodes in ad hoc networks. In: Proc. of the 12th Int. Conf. on Advanced communication technology, ICACT 2010 (2010)
10. Xu, L., Lon, Z., Ye, A.: Analysis and countermeasures of selfish node problem in mobile ad hoc network. In: Proceedings of the Tenth International Conference on Computer Supported Cooperative Work in Design, CSCWD 2006 (2006)
11. Zhong, S., Chen, J., Yang, Y.: Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In: Proceedings of the Twenty-second Annual Joint Conference of the IEEE Computer And Communications Societies, INFOCOM 2003 (2003)

A Distributed Allocation Strategy for Data Mining Tasks in Mobile Environments

Carmela Comito, Deborah Falcone, Domenico Talia, and Paolo Trunfio

Abstract. The increasing computing power of mobile devices has opened the way to perform analysis and mining of data in many real-life mobile scenarios, such as body-health monitoring, vehicle control, and wireless security systems. A key aspect to enable data analysis and mining over mobile devices is ensuring energy efficiency, as mobile devices are battery-power operated. We worked in this direction by defining a distributed architecture in which mobile devices cooperate in a peer-to-peer style to perform a data mining process, tackling the problem of energy capacity shortage by distributing the energy consumption among the available devices. Within this framework, we propose an energy-aware (EA) scheduling strategy that assigns data mining tasks over a network of mobile devices optimizing the energy usage. The main design principle of the EA strategy is finding a task allocation that prolongs network lifetime by balancing the energy load among the devices. The EA strategy has been evaluated through discrete-event simulation. The experimental results show that significant energy savings can be achieved by using the EA scheduler in a mobile data mining scenario, compared to classical time-based schedulers.

1 Introduction

A growing number of mobile data intensive applications appeared on the market in recent years. Examples include cell-phone- and PDA-based systems for body-health monitoring, vehicle control, and wireless security systems. Advanced support for data analysis and mining is necessary for such applications. A key aspect

Carmela Comito · Deborah Falcone · Paolo Trunfio
DEIS, University of Calabria, Rende (CS), Italy
e-mail: {ccomito,dfalcone,trunfio}@deis.unical.it

Domenico Talia
ICAR-CNR and DEIS, University of Calabria, Rende (CS), Italy
e-mail: talia@deis.unical.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 231–240.
springerlink.com

that must be addressed to enable effective and reliable data mining over mobile devices is ensuring energy efficiency, as most commercially available mobile devices have battery power which would last only a few hours. Therefore, data mining tasks in mobile environments should be allocated and scheduled so as to minimize the energy consumption of low-capacity mobile devices.

Only very few studies have been devoted on energy characterization of data mining algorithms on mobile devices [1], but not in cooperative distributed scenarios. We worked in this direction by defining a distributed architecture in which mobile devices cooperate in a peer-to-peer style to perform a data mining task, tackling the problem of energy capacity shortage by distributing the energy consumption among the available devices. Efficient resource allocation and energy management is achieved through clustering of mobile devices into local groups, also termed clusters. Such a cooperative architecture can be seen as a set of requestors, i.e., mobile applications generating data mining tasks to be executed, and a clustered set of resources, i.e., mobile devices characterized by different levels of energy and processing power, where tasks can be executed. To make the most of all available resources, a proper distribution of tasks among clusters and individual devices is crucial. The design and evaluation of such energy-aware (EA) task allocation (or task scheduling) strategy is the main goal of this paper.

The design principle of the EA scheduling strategy is to find a task allocation that prolongs network lifetime by balancing the energy load among clusters. To this end, the EA scheduler implements a two-phase heuristic-based algorithm. The algorithm first tries to assign a data mining task locally to the cluster that generated the execution request, by maximizing the cluster residual life. If the task cannot be assigned locally, the second phase of the algorithm is performed by assigning the task to the most suitable node all over the network of clusters, maximizing this way the overall network lifetime. We characterize the energy consumption of mobile devices defining an energy model in which the energy costs of both computation and communication are taken into account.

The EA approach has been introduced in our previous work [2]. In this paper we propose an extensive evaluation of the EA allocation strategy using a custom discrete-event simulator, which allowed us to assess it effectiveness on a large range of data mining tasks. The experimental results show that a significant improvement can be achieved using our EA scheduler compared to the time-based round-robin scheduler. In details, our algorithm: i) is effective in prolonging network lifetime by reducing the energy consumption, without sacrificing the number of tasks completed; ii) in all the experiments performed, it was able to keep alive most of the mobile devices thanks to its energy load balancing strategy.

The remainder of the paper is organized as follows. Section 2 introduces the reference architecture. Section 3 presents the EA allocation scheme. Section 4 presents the experimental results. Section 5 discusses related work. Finally, Section 4 concludes the paper.

2 Reference Architecture

In a mobile ad hoc network, efficient resource allocation, energy management and routing can be achieved through clustering of mobile nodes. In a clustering scheme, mobile nodes are divided into clusters. Generally, geographically adjacent devices are assigned to the same cluster. Under a cluster-based structure, mobile nodes may be assigned different roles, such as *cluster-head* or *cluster member*. A cluster-head normally serves as the local coordinator for its cluster, performing intra-cluster transmission arrangement, data forwarding, and so on. A cluster member is a non cluster-head node without any inter-cluster links.

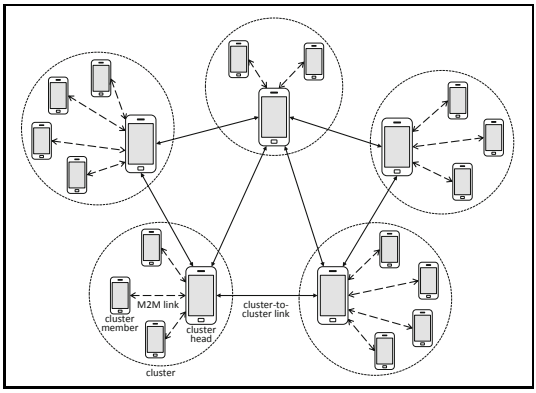


Fig. 1 Reference architecture for mobile-to-mobile collaborations between mobile devices

In this work we assume, as a reference, the cluster-based architecture shown in Figure 1, which is meant to support mobile-to-mobile (M2M) collaborations between mobile devices. Examples of M2M collaborations occur in several domains such as disaster relief, construction management and healthcare. Mobile nodes within a cluster interact through ad-hoc connections (e.g., wi-fi, bluetooth), that we refer to as M2M links, represented as dashed arrows in the figure. Interactions between clusters (cluster-to-cluster links) take place through ad-hoc connections between the respective cluster-heads, and are represented as continuous arrows in the figure.

The architecture is based on a fully distributed cluster formation algorithm in which nodes take autonomous decisions; no global communication is needed to setup the clusters but only local decisions are taken autonomously by each node. This means that the proposed architecture is self-organized into mobile clusters: when mobile devices meet each other, i.e., when they are within the same transmission range, they can form a mobile group. The self-organization nature of the clustering scheme distributes the responsibility between all the mobile nodes. We do not focus in this paper on the cluster formation algorithm as it has been presented in a previous work [3].

All types of interactions in the architecture shown in Figure 1 take place either to ask for a computation request, or to perform a distributed allocation of a data mining task, as detailed in the next section.

3 Distributed Task Allocation Strategy

The energy-aware (EA) scheduling strategy deals with a set of independent data mining tasks, dynamically generated over time, which have to be allocated over mobile nodes organized into the cluster-based architecture introduced earlier.

Task allocation is a step of the more general scheduling problem; it can also be seen as a global scheduling or meta-scheduling that distributes the tasks among the devices. Once tasks have been allocated, the problem becomes one of defining a feasible local schedule that manages task execution for each node. In this paper we focus on the task allocation problem and we refer to task allocation or task scheduling interchangeably.

The task allocation problem has been proven to be NP-Complete in its general form [4]. However, some optimal algorithms have been proposed for restricted versions of the problem and some heuristic-based algorithms have been proposed for the more general versions of the problem allowing to find good allocations in polynomial time [5].

We propose a two-phase heuristic-based, decentralized algorithm. When an assignment decision has to be made for a task, the first phase, referred to as *local assignment* phase, is responsible for local task arbitration: it considers the energy consumption of task execution on the different devices within the local cluster. The algorithm tries to minimize the total consumed energy in the cluster by assigning the task to the device that allows to extend the cluster residual life. If the first phase is not feasible, the second phase, referred to as *global assignment* phase, is responsible for task arbitration among clusters: the task will be assigned to the most suitable device, all over the network of clusters, that maximizes the overall network lifetime.

Some definitions and notations are introduced in the following to support the description of the proposed distributed allocation strategy.

- $PC_i(t)$: processing capacity of device d_i at time t.
- $M_i(t)$: memory availability of device d_i at time t.
- $EEC_i(t_j, s)$: estimated energy consumed for computation by device d_i to process a task t_j over a data set of size s.
- $EET_i(t_j, s)$: estimated energy consumed for communication by device d_i to process a task t_j over a data set of size s.
- $EMC_i(t_j, s)$: estimated memory consumption of device d_i to run a task t_j over a data set of size s.
- $EPC_i(t_j, s)$: estimated processing capacity required by device d_i to execute a task t_j over a data set of size s.
- $RL_i(t)$: residual life of node i at time t, defined as follows:

$$RL_i(t) = RE_i(t)/P_i(t) \tag{1}$$

where $RE_i(t)$ is the residual energy available at node i at time t, and $P_i(t)$ the instantaneous power.

The classical task allocation problem can be reformulated here as the problem of finding the proper task assignment that minimizes the energy dissipated in the system. In other words, we formalize the problem of task allocation as an optimization problem. The aim of the optimization is to maximally extend the life of all the nodes in the network by balancing the load proportionally to the energy of each node. We achieve this goal by iteratively trying to improve a candidate solution. A feasible allocation is optimal if the corresponding group residual life (in case of local assignment) or system lifetime (in case of global assignment) is maximized among all the feasible allocations.

The candidate nodes to which a task t_a could be assigned have to satisfy the following constraints:

1. a node d_i must have enough processing power to perform the task over a data set of size s: $EPC_i(t_a, s) < PC_i(t)$
2. a node d_i must have enough energy to perform the task over a data set of size s: $EEC_i(t_a, s) < RE_i(t)$
3. a node d_i must have enough memory to perform the task over a data set of size s: $EMC_i(t_a, s) < M_i(t)$

During the local assignment phase, a cluster-head, or the set of neighboring cluster-heads in case of the global assignment, will choose the local node, among the ones satisfying the above constraints, that will prolong the life of the corresponding local group by using the following objective function:

$$RL_{LG_j}(t) = \text{Max} \sum_{i=1}^{N_{LG_j}} \alpha_i RL_i(t) \tag{2}$$

where RL_{LG_j} denotes the residual life of local group LG_j, N_{LG_j} is the number of nodes within the local group LG_j, RL_i is the residual life of node i in the group, and parameter α_i takes into account the importance of node i in the local group. The node associated with the maximum value in the objective function will be selected by the cluster-head as candidate node. Note that throughout the experimental evaluation presented in the next section, the parameter α_i is set to 1 thus, all the nodes have the same role within the local group.

If the global assignment phase is activated, the final decision is taken by considering all the candidate nodes proposed by the neighboring clusters. The task will be assigned to the local group that maximizes the life of the whole network:

$$RL_{net}(t) = \text{Max} \sum_{j=1}^{N} \alpha_j RL_{LG_j}(t) \tag{3}$$

where N is the number of groups in the network.

4 Experimental Results

In this section we present an experimental evaluation of the proposed EA scheduler, performed using a custom discrete-event simulator. As a first step, the simulator builds a network composed of 100 mobile devices, and let them grouping into clusters based on the algorithm described in [3]. Then, an initial energy capacity ranging from 3,000 J to 11,000 J is assigned to each device, following a normal distribution. After the initial setup, mobile devices start generating a set of data mining tasks to be executed, which are allocated to the available nodes according to the EA strategy described in the previous section.

To the purpose of our simulation, we characterize the data mining tasks on the basis of the energy required to complete their execution. To this end, we selected three reference data mining algorithms (J48, for data classification; Kmeans, for data clustering; and Apriori, for association rules discovery). Each algorithm was used to analyze a sample dataset (of varying size), using an Android smartphone as mobile device. After each execution we measured the actual energy consumed to perform the task, which was used as input for the simulations.

The simulation aims at studying the behavior of the scheduler with respect to the energy depletion and network lifetime. Accordingly, as performance metrics, we use the *number of alive devices*, the *number of completed tasks*, and the *network residual life* at the end of the simulation. To assess the effectiveness of the EA strategy, we compared its performance with the one achieved by round-robin (RR) scheduling algorithm.

In a first set of experiments, for each reference algorithm (J48, Kmeans, or Apriori), we ran a set of tasks by varying the size of the dataset to be mined from 100 kB to 3.2 MB. Each simulation lasts 30 hours, with tasks that arrive following a Poisson distribution with a frequency $\lambda = 160$ tasks per hour. Figures 2(a), 3(a) and 4(a) show the number of alive devices at the end of the simulation for J48, Kmeans, and Apriori, respectively, using the EA and RR strategies. With all data mining algorithms and dataset sizes, the number of alive devices with EA is greater than (and in a few cases equal to) that of RR. In particular, Figures 2(a) and 3(a) show that there are no alive devices with RR for datasets greater than 200-400 kB using J48 and Kmeans. In contrast, in the same configurations, EA keeps alive a high percentage of the devices. Additionally, we can note that, with EA, the number of alive nodes increases with the dataset size. This is due to the fact that, when the dataset size increases, also the energy required to complete the task increases. Since the EA scheduler does not allocate tasks when the available devices do not have enough energy, this lead to a higher number of alive nodes. It is important to note that the higher number of alive devices ensured by EA compared to RR, is obtained without reducing the number of tasks completed, as shown in Figures 2(b), 3(b) and 4(b). Figure 5(a) shows the network residual life measured at the end of the experiments. The figure confirms that the EA scheduler is effective in prolonging network lifetime compared to the RR algorithm.

In a second set of experiments, for each reference algorithm (J48, Kmeans, or Apriori), we ran a set of tasks with a fixed dataset size (200 kB) but with task

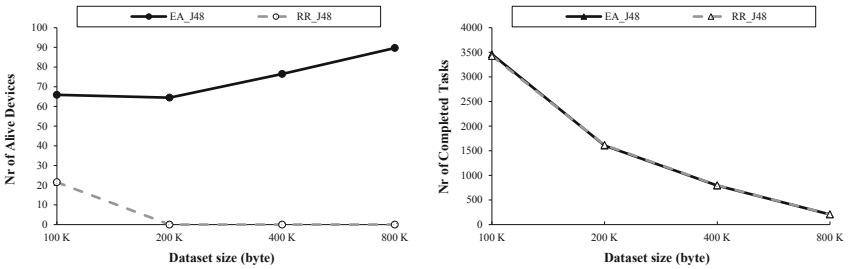


Fig. 2 (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with J48

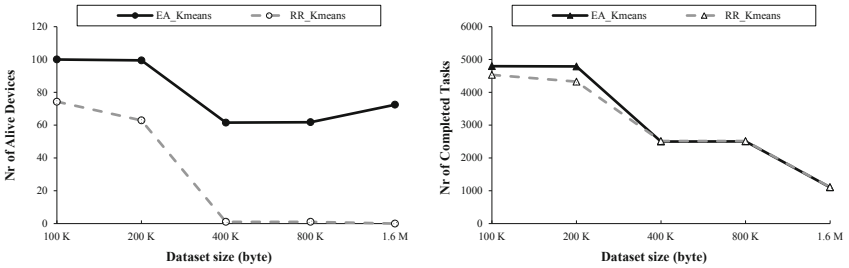


Fig. 3 (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with Kmeans

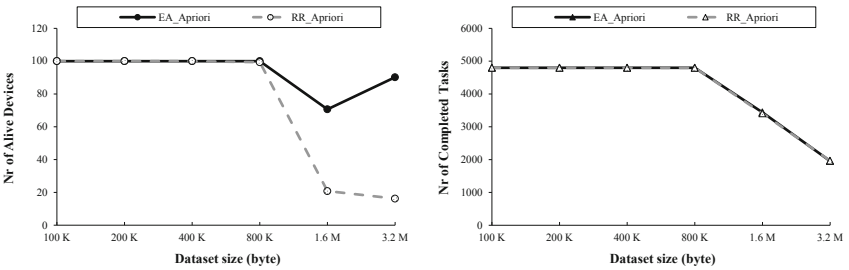


Fig. 4 (a) Number of alive devices and (b) Number of completed tasks w.r.t. dataset size, using EA and RR with Apriori

arrival rate λ varying from 80 to 1280 tasks per hour. Figures 5(b) shows the network residual life measured at the end of the experiments for the three algorithms, using EA and RR. As expected, increasing the task arrival rate, the network residual live tends to zero both for EA and RR. However, for the lightest of the three data mining algorithms (Apriori), the residual life does not reach zero and the

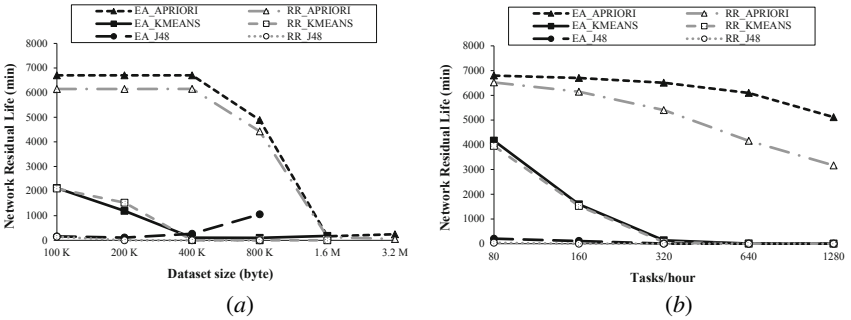


Fig. 5 Network residual life using EA and RR with Apriori, Kmeans and J48, w.r.t. (a) dataset size; (b) task arrival frequency

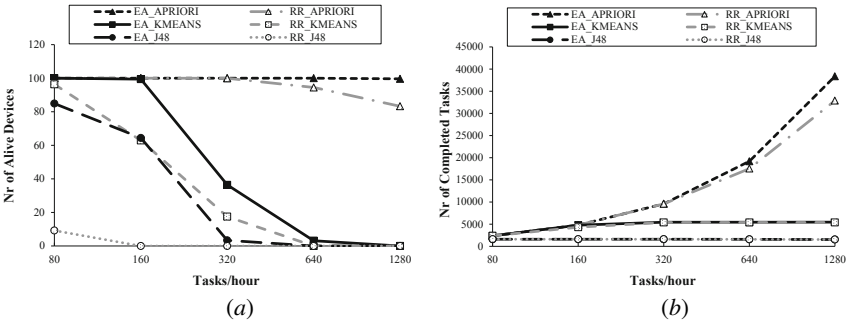


Fig. 6 (a) Number of alive devices and (b) Number of completed tasks w.r.t. task arrival frequency using EA and RR with Apriori, Kmeans and J48

difference between EA and RR increases with λ in favor of EA. Figure 6(a) shows the number of alive devices for the three algorithms, using EA and RR. The results demonstrate, also in this case, that the number of alive devices with EA is greater than that achieved by RR. Figure 6(b) compares the performance of EA and RR in terms of completed tasks for the three algorithms. With Apriori, both EA and RR are able to complete more tasks as λ increases, but EA ensures better performance. With J48 and Kmeans, over a given task arrival rate, the number of completed tasks cannot increase because the network residual life is zero, as shown in Figures 5(b). However, even in this cases, there is a slight advantage for EA compared to RR for $\lambda < 320$ tasks per hour.

The experimental results discussed above demonstrated that a significant improvement can be achieved using the EA scheduler compared to the RR scheduler in a distributed mobile scenario. In details, our algorithm: i) resulted effective in prolonging network lifetime by reducing the energy consumption, without sacrificing the number of data mining tasks completed; ii) in all the experiments performed, it was able to keep alive most of the mobile devices thanks to its energy load balancing strategy.

5 Related Work

Most of the existing research work in the area of energy-aware systems are hardware-based techniques focusing on reducing the energy consumption of the processor. One of the most adopted techniques is turning off idle components [6]. Dynamic Voltage Scaling (DVS) is another technique of energy conservation. DVS refers to the technique of simultaneously varying the processor voltage and frequency as per the energy performance level required by the tasks [7–9]. Remote execution is a software-based technique in which a device with limited energy transfers a computational task to a nearby device which is more energy powerful. Energy-aware task scheduling is another software method where the scheduling policy aims at optimizing the energy.

To the best of our knowledge, little work has been done on energy-aware scheduling over a mobile ad hoc networks (MANETs), and not for data mining scenarios. In [10] an energy-aware dynamic task allocation algorithm over MANETs is proposed. However, this work is different from ours, both for the considered application scenarios and for the underlying architecture and cost function to be optimized. We group the devices in clusters to promote local cooperation among nearby devices and to minimize the transmission energy. This issue is particularly relevant because we have experimentally found that the transmission energy highly impacts on the overall energy consumption. In contrast to ours, the solution proposed in [10] is effective for compute intensive applications and does not address the communication aspects of the system. Furthermore, we adopt a different objective function: we maximize the network residual life rather than minimizing the energy consumption.

Using the residual life parameter we are able to actually consider the real energy consumption rate of single devices, single clusters and the overall network. Conversely, [10] does consider only the local computation issues and it works at a node level ignoring the workload in the rest of the network. Thus, differently from us, they do not take into account the actual load of the devices with the possibility of assigning a task to a device that consumes less energy, but which is less charged compared to another one that consumes more energy but it is more energy powerful and thus could efficiently execute that task.

6 Conclusions

Supporting data mining in mobile environments requires effective architectures and allocation strategies to improve energy utilization of battery-operated devices. We addressed this issue by defining a distributed architecture in which mobile devices cooperate in a peer-to-peer style to perform a data mining process, tackling the problem of energy capacity shortage by distributing the energy consumption among the available devices.

Within this framework, we proposed an energy-aware (EA) task allocation scheme focusing on energy efficiency. To conservatively consume energy and maximize network lifetime the EA adopts a heuristic algorithm that balances the energy

load among all the devices in the network. Experimental results show that significant improvements in terms of residual network lifetime, number of alive devices can be achieved by using the EA scheduler in a mobile data mining scenario, compared to classical time-based schedulers such as round-robin.

Acknowledgements. This work is partially funded by European Commission, European Social Fund (ESF), and Regione Calabria.

References

1. Bhargava, R., Kargupta, H., Powers, M.: Energy Consumption in Data Analysis for On-Board and Distributed Applications. In: ICML 2003 (2003)
2. Comito, C., Falcone, D., Talia, D., Trunfio, P.: Energy Efficient Task Allocation over Mobile Networks. In: IEEE CGC 2011, pp. 380–387 (2011)
3. Comito, C., Talia, D., Trunfio, P.: An Energy-Aware Clustering Scheme for Mobile Applications. In: IEEE Scalcom 2011, pp. 15–22 (2011)
4. Garey, R., Johnson, D.: Complexity Bounds for Multiprocessor Scheduling with Resource Constraints. SIAM J. Computing 4, 187–200 (1975)
5. Chang, H.W.D., Oldham, W.J.B.: Dynamic Task Allocation Models for Large Distributed Computing Systems. IEEE Trans. Parallel Distrib. Syst. 6, 1301–1315 (1995)
6. Li, K., Kumpf, R., Horton, P., Anderson, T.: A Quantitative Analysis of Disk Driver Power Management in Portable Computers. In: Winter 1994 USENIX Conference, pp. 279–292 (1994)
7. Zhuo, J., Chakrabarti, C.: An Efficient Dynamic Task Scheduling Algorithm for Battery Powered DVS Systems. In: ASP-DAC 2005, pp. 846–849 (2005)
8. Zhang, Y., Hu, X., Chen, D.: Task Scheduling and Voltage Selection for Energy Minimization. In: DAC 2002, pp. 183–188 (2002)
9. Aydin, H., Melhem, R., Moss, D., Mejia-Alvarez, P.: Power-Aware Scheduling for Periodic Real-Time Tasks. IEEE Trans. Computers 53(5), 584–600 (2004)
10. Alsalih, W., Akl, S.G., Hassanein, H.S.: Energy-Aware Task Scheduling: Towards Enabling Mobile Computing over MANETs. In: IPDPS 2005, vol. 242a (2005)

Adaptive Patterns for Intelligent Distributed Systems: A Swarm Robotics Case Study

Mariachiara Puviani, Giacomo Cabri, and Letizia Leonardi

Abstract. In order to propose some evaluation of adaptive architectural patterns for intelligent distributed systems, in this paper we present a case study where a swarm of robots is required to coordinate and adapt to perform a task. We will present the results of several simulations and propose some considerations that help further research about adaptive patterns in intelligent distributed systems.

1 Introduction

Nowadays adaptation in intelligent distributed systems, that is defined as the ability of a system to change its behaviour to dynamic operating conditions [4], is a very important aspect for these systems. Adaptation can be applied to intelligent distributed systems at two levels: at the level of a *single component* (components are able to automatically change their behaviour according to the changes in their context) and at the level of a *whole system* (the entire system can exhibit adaptive capabilities). In the latter case adaptation can be achieved in two different ways: (i) by *explicitly* programming each component to manage a specific portion of the global goal; (ii) by achieving a global goal through a new behaviour that the whole system will exhibit even if composed of autonomous components. The latter described adaptation is what happens in *swarm robotics*: each robot acts individually using local information. The global goal of the system is achieved by means of the collective behaviour of all the components involved in the system.

Mariachiara Puviani
DISMI - Università di Modena e Reggio Emilia, Via Amendola 2, Reggio Emilia, Italy
e-mail: mariachiara.puviani@unimore.it

Giacomo Cabri · Letizia Leonardi
DII - Università di Modena e Reggio Emilia, Via Vignolese 905, Modena, Italy
e-mail: {giacomo.cabri,letizia.leonardi}@unimore.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 241–246.
springerlink.com

In this paper, we present swarm robotics' simulations to show how adaptivity comes out in different situations, and to study which adaptive pattern can be useful in different situations. This evaluation helps us to understand how different patterns work and to decide which pattern is more appropriate.

2 Swarm Robotics and Adaptive Architectural Patterns

Swarm intelligence scenarios are an ideal starting point to understand how adaptation works in intelligent distributed systems. In swarm robotics, *task allocation* [2] is a well known problem, where the *goal* of each robot is to search for food items placed in an area, and bring them back to the nest. Each robot has also the *sub-goal* of avoiding obstacles (e.g., walls, other robots or objects) on its way. Thus, each robot needs to change its route in order to avoid obstacles while adapting its behaviour to a diminishing number of available food items. The goal of the system is to increase the nest energy with food items. This energy tends to decrease during the simulation, due to the energy consumption of each robot. The main *constraint* of each robot is to avoid running out of batteries. Running outside the nest, robots are consuming energy taken from the nest itself. Each robot must save its energy in order to keep the whole system up and running.

In this context, the adaptive behaviour of the system can be defined by a pattern. An *adaptive architectural pattern*[1] is a conceptual scheme that describes a specific adaptation mechanism. It specifies how the component/system architecture can express adaptivity. When developing an intelligent distributed system that needs to be adaptive, such as a swarm robotics one, the use of an appropriate pattern that will enact adaptivity will help developers in their work. The pattern permits to the developer to be guided to make the system exhibit a required behaviour, even when unexpected situations occur. Unfortunately, it is not always possible to immediately recognize which is the most appropriate pattern that the developers can use when building intelligent distributed systems. Thus, it is useful to have guidelines that explain the features of each patten, so the developers can choose the one they think is the most appropriate one for their needs. In order to verify if a pattern is the appropriate one for a specific system, we use a "black-box" approach: we implement the system considering when it needs to adapt and under which conditions it will adapt. In this way we can consider, for each case study, the tolerance under which the system can adapt itself during its execution in order to continue to satisfy its goals and constraints. This tolerance describes an area where the system should remain to be considered adaptive.

In [1], we proposed a preliminary list of patterns and offered examples for their use. In this paper, we proceed to understand whether exploiting a specific pattern can be useful to implement an intelligent distributed system. In our swarm robotics case study we apply the pattern that in [1] we called "pattern based on swarm intelligence connected with the environment". This pattern addresses the problem of coordinating a large number of simple components (with limited knowledge) in order

[1] In the following, we use the term *pattern* to mean always *adaptive architectural pattern.*

to achieve a global goal, whose explicit representation is not possible. The collective behaviour results from components' behaviour adjusted by local environment conditions. The single components are not able to directly communicate one with the other, but an implicit communication is made using the *environment* that propagates the adaptation. So the environment plays the role of a strong stimulus for components that aims at modifying their internal dynamical behaviour indirectly.

3 Swarm Robotics Simulations

In order to simulate the robot behaviour for our case study, we use ARGoS[2] [3]. We define a simple arena where robots are free to move (see Figure 2). The robots must bring the food find in the arena, to the nest to increase the nest energy, and there they can also recharge their batteries, decreasing the nest energy.

The behaviour of single robots is probabilistically determined. Each exploring robot returns to the nest with probability *Pr* and starts exploring with probability *Pe*; these probabilities change depending on the situation, and this makes the behaviour of each robot adapt according to the behaviour of all the other robots and to environmental changes (i.e. change in the number of food items, addition of obstacles and so on). Every robot, while behaves in order to satisfy its goal, changes its probabilities. The other robots, sensing the changes that are propagated in the environment, adapt their behaviour in their turn. Doing that they change probabilities again, and make the system to continuously adapt. Each robot in our simulation uses the "pattern based on swarm intelligence connected with the environment" presented in Section 2. In the following, we show how adaptation of the ensemble of robots works under different operational conditions, and if the chosen pattern is always suitable for the studied system.

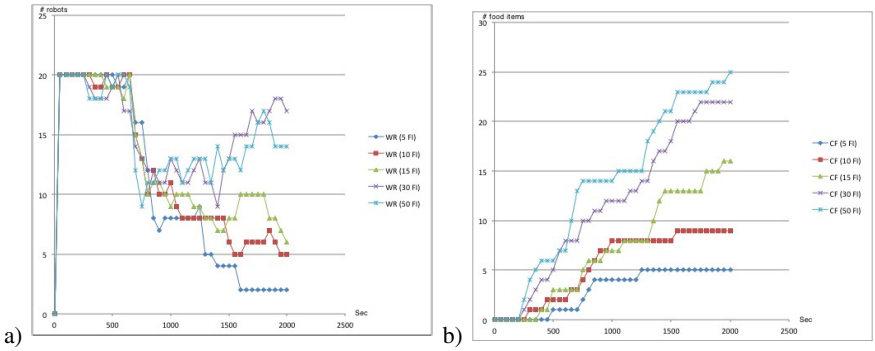
3.1 Changing Number of Food Items

In this simulation, we have a fixed number of robots (20) acting in the arena, and we set a variable number of food items (starting from 5 to 50 food items). We can observe the different behaviour of the system when we change the number of items: robots try to adapt their actions in order to avoid too long unfruitful explorations while trying to collect as much food as possible to increase the nest energy.

Figure 1 shows simulation results, changing the number of food items in the arena. Specifically, Figure 1 (a) shows the number of walking robots (WR) in time, while Figure 1 (b) shows the number of collected food items (CF) in time.

As expected, Figure 1 (a) shows that for the time from 0 to 750 sec, the behaviour of the system is quite the same for all the different scenarios: initially the robots are all out searching for food, and then they start to adapt to the environment. The robots stay out of the nest, looking for food, only if there is a large amount of food around, because their probability of finding food in a short time increases. In this situation,

[2] `http://iridia.ulb.ac.be/argos/`



a) b)

Fig. 1 Walking robots (a) and overall collected food items (b)

they are not losing energy and are motivated to resume exploring, after the pause in the nest, especially when the number of food items is higher than the number of robots. The chance of increasing the nest energy in this case is higher than the use of the energy itself. A robot can spend less than 50% of its energy to find the food and to return to the nest due to the large amount of food. Doing that, it increases the nest energy of more than what it spends. As an example, in this simulation the charged battery costs 1000 to the nest and each food item gives 500. When the food items present in the arena are more than 30, the average of battery consumption for each robot is 400, so there is a constant increase of energy (about 100).

We can see another adaptive behaviour in the system when the number of food items is low (5 Food Items - FI - or 10 FI): robots stay out of the nest for long time because they are more than the number of items in the arena, so their probability to return to the nest (Pr) and stay there rapidly increases.

Another expected result comes from the adaptation of the behaviour of every single robot: the number of collected items grows more rapidly when we have a higher availability of food in the arena (see Figure 1 (b)). That is because robots can quickly find food and their probability of staying out (Pe) for nothing decreases.

As a summarizing consideration, in this simulation the used pattern seems to be the appropriate one because the number of robots is fixed and the environment, that is the means of adaptation, is frequently changing, so it is the best way to propagate adaptation.

3.2 Changing Obstacles

In this simulation, we show how the system of robots adapts in environments with different obstacle locations. The number of robots is fixed (10), as the number of food items (15). Figure 2 shows the three experimental settings we used: the first one with no obstacles, the second with a short wall in the middle of the arena, and the third one with a long wall.

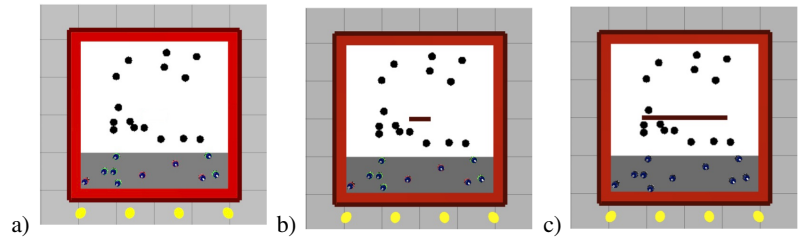


Fig. 2 Simulation arenas. White floor, grey nest and food items represented as black dots

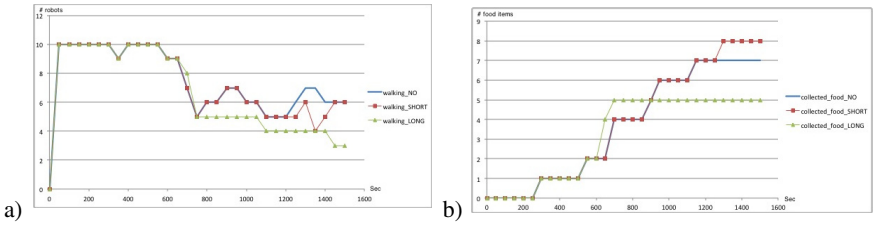


Fig. 3 Walking robots (a). Overall collected food items (b)

We can see from Figure 3 (a), that the number of walking robots is the same for the first part of the simulation (for around 700 sec). After that, the number of robots walking within the arena with a long obstacle sharply reduces. This happens because it is more difficult to find food and to come back to the nest, due to the obstacle. So robots spend more energy to avoid the obstacle and the probability to stay in the nest (Pr) is higher.

The same considerations can be done looking at the collected food (see Figure 3 (b)) in the three different settings. The number of food items that are brought to the nest is less when we have the long obstacle in the arena. It is interesting to observe that the number of items is larger (8 instead of 7) when there is the short obstacle in the arena, with respect to the case in which there is no obstacle at all. In fact, the short obstacle forced the robot to change their path to avoid it, and in this case, the change of path helps in finding the nest way or in finding a new food item.

In this simulation, we see that it can be useful to consider a different pattern than the used one, for the scenario with the long obstacle in the arena. Here a direct communication between robots is more useful to map the environment and help robots in their task: the obstacle makes robots that do not know the environment, frequently change their direction. This makes the robots to consume battery and not to be sure to find food items. An alternative pattern can be the one with a direct communication between robots, e.g. based on negotiation. The possibility to communicate how the external environment is in a given time tick, helps robots to find new food items and to localise the nest.

4 Conclusions

Using an appropriate pattern makes it possible to obtain an *intelligent* adaptive system even starting from components that behave simply in a probabilistic way and that have a limited knowledge about the environment and others components. The simulation reported in this paper suggest that some patterns are more suitable than others to build a specific system because they better specify adaptation mechanisms for the involved components and for the whole system. We shown that the "pattern of swarm intelligence connected with the environment" was the most appropriate for this kind of systems, for the majority of the cases. We have also shown that for some specific situations, this pattern is not the best one, and it would be better to apply another one that better suits the adaptive situation.

Our future work will focus on simulating others patterns to better understand their usage, and on enabling self-expression [5], which is defined as the capability of changing the whole pattern that descibes adaptation when the change of situation may require it (e.g. passing from the "pattern of swarm intelligence connected with the environment" to a pattern based on an "external adaptive manager" when the number of robots rapidly decreases, and the last pattern better manage the system).

Acknowledgements. Work supported by the ASCENS project (EU FP7-FET, Contract No. 257414).

References

1. Cabri, G., Puviani, M., Zambonelli, F.: Towards a taxonomy of adaptive agent-based collaboration patterns for autonomic service ensembles. In: 2011 Int. Conf. on Collaboration Technologies and Systems (CTS), pp. 508–515. IEEE (2011)
2. Krieger, M.J.B., Billeter, J.B.: The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. Robotics and Autonomous Systems 30(1), 65–84 (2000)
3. Pinciroli, C., Trianni, V., O´Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Stirling, T., Gutirrez, A., Gambardella, L.M., Dorigo, M.: ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In: Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2011), pp. 5027–5034. IEEE Computer Society Press, Los Alamitos (2011)
4. Weyns, D., Holvoet, T.: An architectural strategy for self-adapting systems. In: Proc. of the Int. Workshop on Software Engineering for Adaptive and Self-Managing Systems, p. 3. IEEE Computer Society, Minnesota (2007)
5. Zambonelli, F., Bicocchi, N., Cabri, G., Leonardi, L., Puviani, M.: On self-adaptation, self-expression, and self-awareness in autonomic service component ensembles. In: Fifth IEEE Conf. on Self-Adaptive and Self-Organizing Systems Workshops (SASOW), pp. 108–113. IEEE (2011)

Maximal Component Detection in Graphs Using Swarm-Based and Genetic Algorithms

Antonio González-Pardo and David Camacho

Abstract. Nowadays, there is an increasing interest in the application of Collective Intelligence and Evolutive optimization algorithms for solving *NP-complete* problems. This is because the solution or optimization process of these type of problems requires a huge amount of resources (such as computational effort or time). Some examples of these types of problems are scheduling problems, constrained satisfaction problems, or routing problems. Collective strategies are heuristics that allow to look for new solutions in real complex problems using concepts extracted from a metaphor of social behavior of ants, bees, bacteria, flocks of birds and/or schools of fish. In this paper we propose a practical comparison between a classical Genetic-based approach and a Swarm-based strategy applied to the detection of maximal component in graphs. This work describes how these two different optimization strategies can be adapted and used to extract the different sub-graphs that contains the maximum number of nodes. Experimental results show the best results are obtained using ACO algorithm, but new strategies must be taken into account in order to improve the results.

1 Introduction

Past decades have shown a growing interest in Collective Intelligence (CI) [8] and Evolutionary Algorithms (EA) [4, 6]. It has been demonstrated how these optimization algorithms can be applied to wide range of real-world complex problems. Collective algorithms provide a new perspective from other Evolutionary Algorithms, that currently can be considered more classical, like Genetic Algorithms [5], Genetic Programming, or Grammatical Evolution amongst others. However, both kind of algorithms provide an interesting approach based on heuristic search.

Antonio González-Pardo · David Camacho
Computer Science Department, Escuela Politécnica Superior
Universidad Autónoma de Madrid.
C/Francisco Tomás y Valiente 11, 28049 Madrid, Spain
e-mail: {antonio.gonzalez,david.camacho}@uam.es

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 247–252.
springerlink.com

Collective-based Intelligence strategies allow to look for new solutions using concepts extracted mainly from a metaphor of social behavior in sets of self-organizing communities, like ants [1, 3], bees[7] or bacterias [2] amongst others, where the iterations among individuals generate collective knowledge. In these algorithms the members of the population travel through the solution space in order to obtain the best solution to the problem.

This is the main different between Collective-based Intelligence and EA approaches, where each individual is evaluated by a fitness function that allows the comparison between different individuals. This evaluation allows to select the better individuals for the generation of the next population using the *crossover* and *mutation* operator.

There are several real-world complex problems where the task of obtaining a solution needs a huge amount of resources (such as computational effort or time) due to the complexity of the problem. Some examples of these type of problems (usually called *NP-complete* or *NP-hard*) are *scheduling problems*, *constrained satisfaction problems*, or *routing problems*.

This paper presents a case study on the application of classical Genetic-based approach and a Swarm-based strategy in *NP-complete* or *NP-hard* problems. These algorithms are applied to the detection of maximal connected components of an undirected graph. A connected component is a subgraph in which any two vertices are connected to each other by paths. Although this problem can be solved using some linear-time algorithms, this work is useful to understand the strengths and weaknesses of these algorithms applied to *NP-complete* problems.

2 The ACO Approach

The first algorithm studied in this work is an Ant Colony Optimization algorithm (ACO) [3]. In this case, the agents (ants) travel through the network trying to visit all the nodes.

In order to do that, each ant starts its execution in a random node and it travels through the network until all the neighbours of the visited node have been visited by the ant.

Initially, given a specific node, its neighbours have the same probability for being selected. Nevertheless, ants communicate with each other through the environment using *pheromones*. These pheromones are deposited by ants when they make a decision and influences on the decision of other ants. Given an ant located in node i, the probability of travelling from node i to its neighbour j is computed by Eq. 1.

$$p_{i,j} = \frac{\tau_{i,j}^{\alpha} * \rho_{i,j}^{\beta}}{\sum_{l \in N_i} \tau_{i,l}^{\alpha} * \rho_{i,l}^{\beta}} \tag{1}$$

Where $\tau_{i,j}$ is the pheromone concentration in the edge connecting node i with node j, $\rho_{i,j}$ is the value of the heuristic function, and N_i is the set of nodes connected to node i.

Usually, ACO has been applied to the Travelling Salesman Problem (TSP) and the heuristic function is usually based on the distance between the nodes. In this work, as the distance does not matter the heuristic function only takes into account those unvisited nodes. In Equation 1 the value for $\rho_{i,j}$ is 1 if the node has not been visited, and 0 otherwise.

The process of depositing a pheromone on an edge of the graph is performed in two phases. In the first one, a fixed quantity of pheromone is deposited on the edge of the graph. This quantity is defined in a configuration xml file. The second phase starts, when the ant reaches a node i with all its neighbours visited. In this case the ant is trapped, and it goes back in the followed path updating the pheromones. As the goal of the work is to discover the Maximal Component of the graph, the larger path is, the better solution is found. For that reason, the quality of a path is the percentage of nodes included in the component. The values for this function go from 0 to 1, where 0 means that the algorithm did not find any connected component and 1, that the algorithm has found the maximal connected component.

Finally, there is a *stigmergy* process that simulates the pheromone evaporation process, this process it is executed in each iteration and it is used to forget wrong path discover by the ants.

3 GA Approach

The Genetic Algorithm applied in this work, initializes individuals with a random phenotype length. The maximum length of the phenotype is the number of nodes of the graph and each gene contains the name of a node selected randomly from the graph. The phenotype represents a possible connected component that will be evaluated against the graph.

This evaluation provides a fitness value that will be used to select the better individual in a specific population to generate the new population. Phenotypes are evaluated with the same function used in ACO algorithm for updating the pheromones (i.e. the value of a phenotype is the percentage of nodes that compose this connected component). Nevertheless, there are some key concepts that must be highlighted.

The evaluation algorithm starts visiting the first node in the phenotype, and gets its neighbours. From this set, those nodes that have been visited are discarded, and from the remaining nodes one node is selected randomly taking into account that those nodes that belong to the phenotype and they have not been visited yet, have higher probabilities to be selected. The selected node represents the new node that belongs to the current connected component. The evaluation finishes when the algorithm reaches a node and all its neighbours have been visited or they do not belong to the phenotype.

The generation process is composed by two different operators: crossover and mutation. The first one is used to produce new individuals based on the phenotypes of two selected parents. For each parent a random crossover point is selected and their corresponding parts are interchange to produce the new individual. It is

important that if a new individual has to inherit a specific node from both parents, the individual will only inherit one of then (in this case the father's gene). The second operator it is used to escape from local optima. The mutation operator changes randomly a specific gene based on a specific probability.

4 Experiments

This section shows the experimental results over different graph topologies using the previously described algorithms. Two different graph topologies have been taken into account: random graph and small world graph.

Random graphs, or Random networks, are networks where the creation of edges depends on an probability (p). Small World graphs, or Small World networks, are studied because is the topology most commonly used in communication networks due to its characteristic. In this topology each node has a maximum number of connections defined by the connectivity degree (k) and these connections are redirected according to the redirection probability (p).

The goal of the first set of experiments is to compare the number of different connected components discovered by each algorithm. Each experiment is executed in graphs composed by 10 nodes during 10 generations or iterations, and it has been repeated 10 times. The rest of parameter values and the number of different connected components are shown in Table 1.

Table 1 Number of different connected components and the parameter values for the first set of experiments. Each experiment has been executed in graphs composed 10 nodes, the number of generations is 10 and each experiment has been repeated 10 times.

Algorithm	Population Size	Graph Type	Connectivity Degree	Probability	Different Component
ACO	5	Small World	1	0.1	21
	5			0.9	25
	10			0.1	14
	10			0.9	28
	5	Random	-	0.1	10
	5			0.9	39
	10			0.1	10
	10			0.9	74
GA	5	Small World	1	0.1	87
	5			0.9	124
	10			0.1	114
	10			0.9	163
	5	Random	-	0.1	25
	5			0.9	549
	10			0.1	18
	10			0.9	1093

As can be seen in Table 1 the genetic algorithm finds more different connected components. This is not exactly true, because some connected components are included into bigger ones. It means that for example, GA finds a connected component that includes node N_A and node N_B, but it finds also other component that contains N_A, N_B and N_C so the first one is included into the second one. On the other hand, ACO algorithm does not show this problem, and for this reason the rest of the experiments are carried out using this algorithm and ignoring GA.

The number of edges that composed the graph affects to the performance of the ACO. The higher number of edges, the slower the algorithm will find the connected components because the number of different paths is higher. This study has been performed using a Small World graph composed by 100 nodes and the redirection probability is fixed to 0.15. In 30 executions, the connectivity degree of the graph is 5 and in other 30 executions this parameter is set to 10. In all the execution, ACO algorithm composed by 50 ants is executed during 100 iterations. The results showed in Figure 1a show that if the number of connection increases, ACO algorithm will need more iterations to find the connected components.

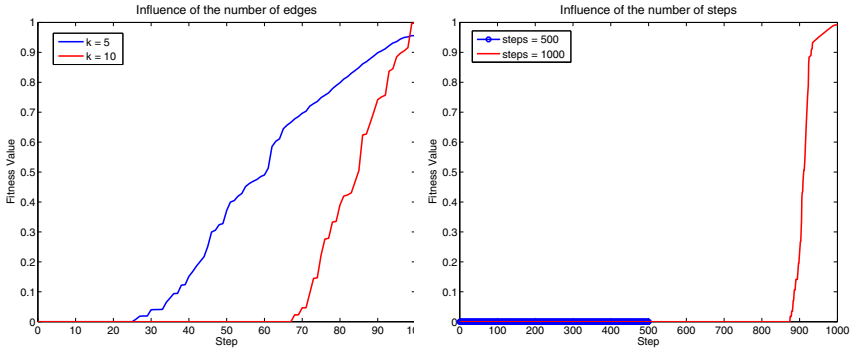


Fig. 1 Left figure represents the influence of the number of edge in the performance of the ACO. 50 ants are executed during 100 steps in a Small World graph composed by 100 nodes and a redirection probability set to 0.15. Right figure shows the evolution of the fitness value of the ACO in a Small World graph is composed by 1000 nodes with a connectivity degree of 50 and the redirection probability is set to 0.15. This ACO has 100 ants. All the experiments are repeated 30 times.

Finally, Figure 1b shows how the number of steps (that the algorithm is running) influences to the performance of the algorithm. A step is a simple execution of all the ants that compose the ACO. For this experiment, 100 ants are executed in a Small World graph composed by 1000 nodes, with a connectivity degree of 50 and a redirection probability set to 0.15. As can be seen in this figure, the number of steps influence the performance of the algorithm because with 500 steps the algorithm does not find any component.

5 Conclusions

This paper studies how Genetic Algorithm and Ant Colony Optimization can be applied to solve *NP-complete* or *NP-hard* problems.

Genetic Algorithm discovers more connected components than Ant Colony Optimization algorithm. This is due to GA generates some partial solutions that are contained into other solutions. This problem can be avoided by using some grouping algorithms that reduce the number of solutions found by the GA.

On the other hand, ACO algorithm provides better results than GA and the results converge faster to the solution due to *pheromones* guide the search process. Nevertheless, this algorithm discovers only continuous connected components without any branch. This is produced because ants communicate with each other using pheromones and the information that they transmit is that "someone" has taken this specific path but ants do not propagate their own information. This problem force to set the number of steps equal or greater than the number of nodes, because the only way of discovering the maximal connected component of a graph is visiting all the nodes of the graph.

As a future work, the design of new *cooperative ants* that share their partial solution will be studied in order to reduce the number of iterations needed to solve the problems.

Acknowledgements. This work has been supported by the Spanish Ministry of Science and Innovation under grant TIN2010-19872 (ABANT).

References

1. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed optimization by ant colonies. In: European Conference on Artificial Life, pp. 134–142 (1991)
2. Das, S., Biswas, A., Dasgupta, S., Abraham, A.: Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications. In: Abraham, A., Hassanien, A.-E., Siarry, P., Engelbrecht, A. (eds.) Foundations of Computational Intelligence Volume 3. SCI, vol. 203, pp. 23–55. Springer, Heidelberg (2009)
3. Dorigo, M.: Ant colony optimization: A new meta-heuristic. In: Proceedings of the Congress on Evolutionary Computation, pp. 1470–1477. IEEE Press (1999)
4. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2009)
5. Fogel, D.B.: Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press (1995)
6. Jong, K.A.D.: Evolutionary computation a unified approach. MIT Press (2006)
7. Karaboga, D., Basturk, B.: A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. J. of Global Optimization 39, 459–471 (2007)
8. Ridge, E., Curry, E.: A roadmap of nature-inspired systems research and development. Multiagent Grid Syst. 3, 3–8 (2007)

How to Monitor QoS in Cloud Infrastructures: The QoSMONaaS Approach

Giuseppe Cicotti, Salvatore D'Antonio, Rosario Cristaldi, and Antonio Sergio

Abstract. Cloud computing provides a new paradigm to enable on-demand delivery of computing resources such as infrastructure, platforms and software as utilities to customers. When moving applications and/or data in the cloud, numerous challenges exist in leveraging the full potential that cloud computing promises. In this paper we address the challenging issue of QoS monitoring in a cloud environment. The availability of a dependable (i.e. reliable and timely) QoS monitoring facility is key for the real take up of cloud computing, since - by allowing organizations to receive the full value of cloud computing services - it would increase the level of trust they would place in this emerging technology. In this paper, we present a dependable QoS monitoring facility which relies on the "as a Service" paradigm, and can thus be made available to virtually all cloud users in a seamless way. Such a facility is called QoSMONaaS, which stands for "Quality of Service MONitoring as a Service". Details are given about the internal design and current implementation of the proposed facility.

1 Introduction

Cloud computing is the new trend of computing where readily available computing resources are exposed as a service. These computing resources are generally offered as pay-as-you-go plans and hence have become attractive to cost conscious customers. Due to the dynamic nature of cloud computing environments, the development of tools for continuous monitoring of the Quality of Service (QoS) associated

Giuseppe Cicotti · Salvatore D'Antonio
University of Naples Parthenope, Naples, Italy
e-mail: {giuseppe.cicotti,salvatore.dantonio}@uniparthenope.it

Rosario Cristaldi · Antonio Sergio
Epsilon srl, Pozzuoli, Italy
e-mail: {rosario.cristaldi,antonio.sergio}@epsilonline.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 253–262.
springerlink.com

to a cloud-based service instance is a complex task. From a technical perspective, cloud computing makes QoS monitoring extremely challenging, and in particular:

- Business applications currently run (mostly) on privately owned service infrastructures, whose technology is - to a large extent - known. In a cloud computing context, they run on infrastructures which are: (i) shared, and (ii) virtualized. This also applies to the QoS monitor itself.
- It has to cope with dynamic composition of services, meaning that it must be able to track the evolution of services and it must be able to provide continuous QoS monitoring, i.e., even while changes are taking place.

Furthermore, since demands of the service consumers vary significantly it is not possible to fulfill all consumer expectations from the service provider perspective and hence a balance needs to be made via a negotiation process. At the end of the negotiation process, provider and consumer commit to an agreement. This agreement is referred to as a Service Level Agreement (SLA). This SLA serves as the foundation for the expected level of service between the consumer and the provider. The QoS attributes that are generally part of an SLA (such as response time and throughput) however change constantly and to enforce the agreement, these parameters need to be closely monitored. We claim that QoS monitoring should be made available to all cloud users in a seamless way. To this end, the ”as a Service” model stands out as the ideal solution. Thus, we decided to implement a QoS monitoring facility which is provided according to the as a Service paradigm. Such a facility is called QoSMONaaS, which stands for ”Quality of Service MONitoring as a Service”. QoSMONaaS has been designed and developed in order to meet the following main requirements:

- the ability to work and interoperate in a heterogeneous and dynamic Cloud Computing environment, providing a simple and clear web-service interface,
- the separation between the monitoring process and the monitored system, while still guaranteeing an impartial evaluation of QoS and the non-repudiation of detected violations,
- high performance by transferring time-consuming tasks like event collection, event-pattern recognition and event correlation to a Complex Event Processor (CEP).

The remainder of this paper is structured as follows. Section 2 presents the QoSMONaaS approach to SLA monitoring, while the architecture of the QoSMONaaS framework is described in section 3. In section 4 the QoSMONaaS operation is illustrated and details about the QoSMONaaS tasks are provided. Implementation details are given in section 5. Section 6 presents related work in the field of QoS monitoring. Finally, section 7 provides some concluding remarks.

2 The QoSMONaaS Approach to SLA Monitoring

An SLA is a contract between the consumer and the provider of a specified service. SLAs are signed upon subscription and are usually prepared from templates specifically conceived for the available services. Templates are used during negotiation to

define the required level of service and they simplify the negotiation process. The user, in fact, might ignore the details of the service he is willing to demand, either because such information is not available at all, or because he lacks the necessary technical skills required to understand their semantics.

SLAs may be considered as formed by two different parts, one containing information that does not depend on the particular service (e.g. user authentication module, information about availability/reliability of the service, privacy and security aspects, etc.) and the other containing service-specific data. In particular, an SLA specifies the Key Performance Indicators (KPI) and the time interval during which they will be measured. In this paper we present the QoSMONaaS platform for QoS monitoring provided according to the "as a Service" paradigm. It presents several innovative aspects with respect to other QoS monitoring systems. Firstly, most existing products only focus on monitoring low-level QoS parameters (network and/or cloud resources), while QoSMONaaS focuses on the performance delivered at the Business Process level, which is what cloud users really care for. Secondly, QoSMONaaS is a flexible monitoring platform, which can be easily configured and customized according to the features of the service being monitored and the needs/preferences of the Service Consumer (SC). Finally, while existing solutions for QoS monitoring have been developed as Trusted Third Parties, thus relying on the assumption that they provide a trusted service, QoSMONaaS is a sort of peer among peers, which will be trusted "by design" (instead of by assumption) by SCs.

QoSMONaaS introduces the concept of Quality Constraint (QC), which expresses a boolean condition for a single KPI. QoSMONaaS allows for the specification of a QC by a Propositional Interval Temporal Logic including only two temporal operators, i.e. *along* and *within*. The statement "P *along* T", where P is a QC and T is a time interval, means that P is true in any time instant belonging to T, while "P *within* T" means that, there exists at least a time instant *i* belonging to T in which P is true The *along* and *within* operators implement the time-based version of the, respectively, "globally" (*G*) and "eventually" (*F*) operators defined in the Linear Temporal Logic (LTL). Therefore, the resulting logic that we take for defining QCs has at least the same expressiveness as the LTL logic fragment that includes only the *G* and *F* operators. It should be noted that QCs without temporal operators are interpreted as expressions to be verified throughout the monitoring validity time defined in the SLA. In such a way, *safety* properties, i.e. something wrong that should never happen, can be easily asserted. The proposed logic can be extended by adding other operators belonging to the linear-time temporal framework in order to increase the expressiveness of the language used to specify QCs.

A QC is specified within the `GuaranteeTerm` section of the SLA. Fig. 1 shows an XML (eXtensible Markup Language) file describing a GuaranteeTerm. In this example the GuaranteeTerm `DowntimeForADay` related to the Metering service is classified as Obligated, which means that the Service Provider (SP) must fulfill it. The QC is formulated by specifying the `KPIName`, that declares the KPI on which the QC has to be verified, and the `Target`, that is the propositional temporal expression to be checked with respect to the specific KPI. In QoSMONaaS KPIs

```
<wsag:GuaranteeTerm Name="DowntimeForADay" Obligated="Provider">
 <wsag:ServiceScope>
  <wsag:ServiceName>Metering</wsag:ServiceName>
 </wsag:ServiceScope>
 <wsag:QualityConstraint>
  <wsag:validityTime frequency="Day" star="01-01-2012,00:00:00"
End ="31-12-2012,23:59:59">
  </wsag:validityTime>
 </wsag:QualityConstraint>
 <wsag:ServiceLevelObjective>
  <wsag:KPITarget>
    <wsag:KPIName>Downtime</wsag:KPIName>
    <wsag:Target> < 2h along 24h</wsag:Target>
  </wsag:KPITarget>
 </wsag:ServiceLevelObjective>
</wsag:GuaranteeTerm>
```

Fig. 1 An example of GuaranteeTerm

involved in QC specification are described by means of an ontology in order to best correlate information about the service being monitored, its quality and the SLA content.

3 QoSMONaaS System Architecture

In this section details about the design of the QoSMONaaS framework are provided and the system architecture is presented. Fig. 2 presents the use case diagram which has been produced while designing QoSMONaaS.

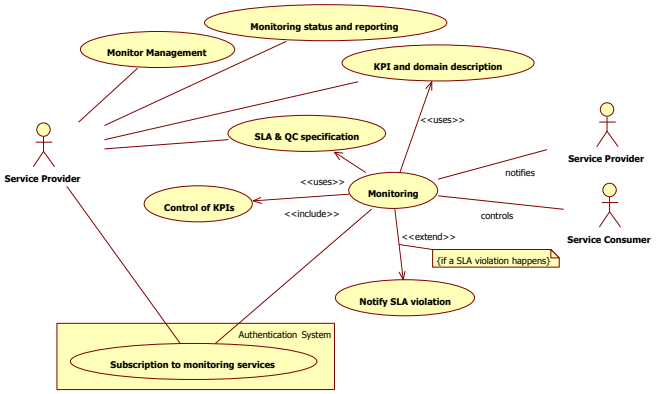


Fig. 2 QoSMONaaS Use Cases

The ”Monitor management” use case refers to the management of the monitoring service. The SC can start, stop and configure this service without any support from the SP. The monitoring facility allows the subscriber to check whether the SP is providing the negotiated level of quality, which has been specified in the SLA, and to obtain reports on the monitored service. The ”Monitoring status and reporting”

use case describes this kind of actions. The ”Control of KPIs” use case describes the actions related to the measurement of the service parameters that have been considered as KPIs. This use case is strictly linked to the use case regarding the actions, which have to be performed in case the results of the monitoring task are different from the expected ones. An important QoSMONaaS functionality regards the notification of an SLA violation. The SC is promptly informed in case a violation of the QCs specified in the SLA occurs. The SC can choose the mean that will be used by the SP to send the notification. These issues are addressed by the ”Notify SLA violation” use case. QoSMONaaS has been designed and developed as a two-tier system architecture, which is composed of a Business logic layer and a Data layer. As shown in fig. 3, the business logic comprises two independent applications, namely the QoSMonitor and the QoSChecker.

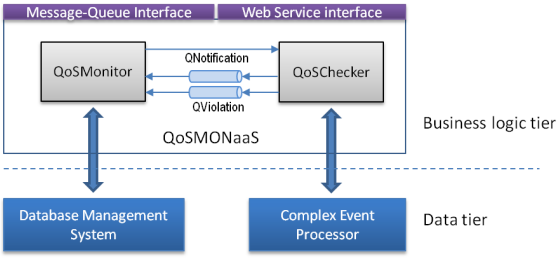


Fig. 3 QoSMONaaS Architecture

On the other hand, the Data tier includes a CEP, which is in charge of processing and analysing real-time data streams, and a DBMS responsible for storing and managing data related to the SLA negotiation, registration, and monitoring.

The QoSMonitor application provides the basic functions of QoS monitoring and SLA violation notification. It can be configured and managed through a proper function which allows for tuning monitoring process parameters. The QoSChecker, instead, is in charge of submitting queries to the CEP in order to monitor selected KPIs.

The QoSMonitor comprises the following components:

Controller. It is the central component whose role is to coordinate the other components in order to drive the monitoring process. It controls the input acquisition, supervises the translation of QCs into queries to be submitted to CEP and is responsible for starting and stopping the SLA monitoring process.

DBManager. It produces a high-level abstraction of the QosMONaaS database schema, thus providing general operations for storing and retrieving data related to SLA negotiation, instantiation, and monitoring.

Parser. It is composed of two distinct sub-parsers. The first sub-parser is the OntologyParser, which is responsible for parsing the ontology describing the negotiated service and the KPIs defined by the subscribers. The second sub-parser is the SLAParser, which is in charge of checking the syntactical correctness of the XML-based

SLA document. The two sub-parsers produce a tree-based representation of the service and of the SLA, respectively, useful for separating front-end technological details from information related to QoSMONaaS back-end components.
Translator. The main task of this component is to extract one or more QCs from the SLA and translate them into one or more queries written in the specific CEP language. The Translator also reads the ontology for retrieving KPIs definition. This component takes care of addressing the different levels of expressiveness characterising the language used to specify the QCs and the language used to formulate CEP queries. It is in charge of translating expressions of the former language into expressions of the latter. It is the only QoSMONaaS component which would be affected in case changes were made to either language. Anyhow, the Translator is designed as an abstract and general class from which a hierarchy of specialized classes can be defined for specific query languages.
Certifier. This component modifies the output of the QoSChecker component by adding a timestamp and a digital signature in order to produce an unforgeable report usable for forensics purposes.

QoSChecker system comprises two components:

QoSDetector. It executes the runtime monitoring algorithm. As soon as a QC violation occurs the QoSDetector inserts the related information in the proper queue named ”QViolation”, which will be read by the QoSManager;
QoSManager. It is in charge of communicating with the QoSMonitor application in order to receive new request for QC verification and send back reports and notification, in case a QC violation happens. The QoSManager manages a pool of QoSDetectors, each of them instantiated as an independent thread performing the monitoring algorithm applied to a single QC.

4 The Dynamic Behaviour of the QoSMONaaS Facility

In this section we describe the sequence of actions performed by the SC and the SP while using QoSMONaaS during the SLA monitoring process. As first step (see fig. 4), the SP and SC register to the Authentication/Anonymization System which will store their real identities and will assign them anonymous credentials (step 1). These anonymous identities are used by the SP and the SC to subscribe to the monitoring service and to allow QoSMONaaS to identify different subscribers without knowing their real identities (step 2). The subscription process requires as input the ontology containing the service and KPI description, and the SLA document. These inputs will be used by QoSMONaaS to infer the QCs and other information needed to perform the SLA monitoring and verification. Then, QoSMONaaS parses and verifies (step 3 and 4) the received input, translates QCs into CEP-specific queries (step 5) and, if no error occurs, sends the generated queries to the CEP (step 6). If the monitoring request is accepted, then the subscriber can start monitoring its SLA (step 7).

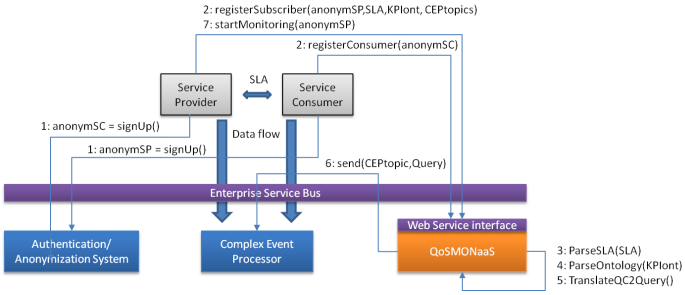


Fig. 4 QoSMONaaS operation

While performing SLA monitoring, QoSMONaaS executes two main tasks:

- a batch process that deals with parsing the input (i.e. the ontology and the SLA document) and translating QCs into CEP-related queries;
- an iterative process, i.e. a continuous interaction between the QoSChecker and the CEP aiming at verifying whether the QCs are met or not.

The batch process includes the following three phases:

Parsing: this phase is responsible for reading inputs and verifying their syntactical correctness. It produces two distinct parse trees, one for the ontology representing the KPIs and another one for the SLA.
Input-Output Analysis: this task extrapolates the QC definitions from the GuaranteeTerm element of the SLA to be monitored together with other information, such as the party obligated to honor the QCs, the priority level of QCs, and the time interval during which the QCs have to be met.
QC-to-Query Translation: this task translates QCs into queries executable by the selected CEP. From each QC the translation task generates a QC-query and one or more KPI-queries. A KPI-query is processed and executed by the CEP in order to get the value of a single parameter or metric, while the QC-query is used to verify whether a set of conditions applied to one or more KPIs are fulfilled. KPI-queries are a fast and simple tool for verifying quality conditions and are applied to prevent events like CEP overload. In case correlation among different conditions is required for SLA monitoring, then QC-queries are submitted to the CEP.

The iterative process consists of the following tasks:

QCs Checking: it verifies whether the QCs are met and produces the output of the quality verification for the service being monitored.
Notification: it generates the customized notification reports according to the consumer's preferences and profile.
Certification: it certifies the generated notification in order to produce an indisputable evidence of the results of SLA monitoring activity. Each certificate includes: the QC verification result, a timestamp (that expresses the time instant when the

certificate is generated), the subscriber's identity (to prove who is the owner of the certificate). The certificate is signed by QoSMONaaS in order to authenticate the source of data.

5 Implementation Details of the QoSMONaaS Monitoring Process

QoSMONaaS has been developed as a monitoring tool capable of working in a event-based environment thanks to the use of a message-queue communication system like Java Message Service (JMS). During the QoSMONaaS operation the following interactions take place:

- between QoSMONaaS and CEP for submitting queries and getting results,
- between QoSMONaaS and SP/SC for requesting the monitoring service,
- between QoSMONaaS and Subscriber to notify SLA violations,
- between monitored service and CEP for exchanging user-related data and events.

All these interactions have been implemented by means of asynchronous communications realized through a publish/subscribe messaging pattern in which logical channels or topics are published by data producers and subscribed by data consumers. Furthermore, in order to optimize the use of the QoSMONaaS facility by SP and SC we exposed the QoSMONaaS API as a web service interface, which includes the following routines:

```
- clientSubscription, providerRegistration
- startMonitor, stopMonitor
- getViolations, SLAstatus, SLAupdate
```

These are the basic functions that are used by an SP and SC in order to subscribe to the monitoring service, get updated information about the SLA status and manage the overall QoS monitoring process. In more details, the SP can register to QoSMONaaS by calling the `providerRegistration()` routine and specifying, among other parameters, the SLAs it wants to offer. Afterwards, the Provider can modify an SLA by invoking the `SLAupdate()` routine. Once the SC agreed with the SP on the level of service to be provided and the SLA has been signed, the SC can require to monitor this SLA through the `clientSubscription()` function. Inputs to this function are the Consumer's and Provider's credentials, the SLA identifier and the type of notification the SC wants to receive in case any violation occurs. After the subscription phase, other utility routines are enabled in order to allow the Subscriber to start/stop the monitoring process (`startMonitor()`/`stopMonitor()`), obtain information on the detected violations with respect to a specific QC (`getViolations()`), and check the status of the registered SLA (`SLAstatus()`).

6 Related Work

Many research activities have been conducted on QoS runtime monitoring. The literature comprises a lot of papers that propose different approaches to QoS monitoring, from solutions relying on Web Services [4, 6, 7] to the more general and wider concept of *aaS in a Cloud environment [2, 3, 5]. Wang et al. [7] propose an online monitoring approach in order to monitor and control QoS of Web Services. They define both a quality model and a middleware-based architecture. The model focuses on what should be monitored by classifying various external/internal events based on their impact on the QoS. On the other hand, the architecture deals with how to monitor QoS by using distributed probes, an independent agent that collects and processes events, and a central analyzer responsible for assessing the QoS level. In [6] QoS monitoring and diagnosis are combined into a single Service Level Management platform, even though the two tasks are offered as independent services. The former is responsible for collecting statistics on the status of the service being monitored and its resources, while the latter is responsible for detecting anomalous parameter values. Both active reporting and passive polling are used as monitoring techniques. In [4] a QoS monitoring system for detection of SLA violations in a service-oriented environment is presented. The authors propose an improved approach that combines both client-side and server-side techniques for SLA monitoring. The proposed system is composed of a QoS monitoring tool on the server-side - integrated with the service to be monitored - and an independent QoS monitor on the client-side - integrated into a runtime environment with extended functionalities (e.g. service selection/composition, event processing, etc). LoM2HiS is a framework proposed by Emeakaroha et al. in [2] as a first step to manage the whole lifecycle of self-adaptable Cloud services. The framework allows for mapping low-level resource metrics onto high-level SLA parameters so as to drive an automatic SLA management module. The proposed framework relies on a communication model based on queuing networks in order to solve potential scalability issues. In [3] the authors present a SLA-based QoS assurance software for a Cloud computing environment that, unlike QoSMONaaS, relies on an independent SLA Monitor implemented by a trusted third-party service provider. Similarly to our approach, in [5] the authors present a Service Management Layer (SML) along with a Monitoring Service (MS) based on the SaaS paradigm. They define an architecture that separates the management layer from the monitoring one in order to implement a loosely-coupled system. The SaaS monitoring system is a low-level monitoring system provided by a third party, while the MS is conceived as a SLA-aware monitor which collects and analyses fine-grain information about the service to be monitored.

7 Conclusions

Organizations are becoming increasingly interested in leveraging cloud computing services to improve flexibility and scalability of the IT services delivered to end-users. However, organizations using cloud computing services face the following challenge: decreased visibility into the performance of services being delivered to

their end-users. Many cloud providers offer dashboards for tracking availability of their services as well as alerting capabilities for identifying service outages in a timely manner, but these capabilities are not sufficient for end-users who need to have a full control of the performance of cloud services in use. More importantly, organizations can not rely on monitoring capabilities offered by their cloud service providers, and they need to deploy third-party solutions that allow them to monitor the performance and levels of SLA achievements of cloud services. We claim that QoS monitoring should be made available to all cloud users in a seamless way. To this end, the "as a Service" model stands out as the ideal solution. Thus, we presented a novel approach to QoS monitoring, and implemented a QoS monitoring facility which is provided according to the as a Service paradigm. Such a facility is called QoSMONaaS, which stands for "Quality of Service MONitoring as a Service". We have illustrated how we have implemented this approach within a Cloud environment. The paper provides two important contributions. First, it proposes an innovative approach to QoS monitoring, which perfectly suits the cloud computing context. Second, it shows how such an approach can be efficiently implemented in a cloud computing platform which provides advanced features, specifically Complex Event Processing (CEP). Future work will aim at developing a commercial grade implementation of QoSMONaaS, which we will use to perform a thorough experimental campaign, with the objective of fully validating the proposed approach.

References

1. Alves, M.C.B., Drusinsky, D., Michael, J.B., Shing, M.-T.: Formal validation and verification of space flight software using statechart-assertions and runtime execution monitoring. In: 6th Int. Conf. on System of Systems Engineering (SoSE), pp. 155–160 (2011), doi:10.1109/SYSOSE.2011.5966590
2. Emeakaroha, V.C., Brandic, I., Maurer, M., Dustdar, S.: Low Level Metrics to High Level SLAs - LoM2HiS Framework: Bridging the Gap Between Monitored Metrics and SLA Parameters in Cloud Environments. In: HPCS, pp. 48–54. IEEE (2010)
3. Hammadi, A.M., Hussain, O.: A Framework for SLA Assurance in Cloud Computing. In: 26th Int. Conf. on Advanced Information Networking and Applications Workshops, pp. 393–398 (2012), doi: 10.1109/WAINA.2012.280
4. Michlmayr, A., Rosenberg, F., Leitner, P., Dustdar, S.: Comprehensive QoS monitoring of Web services and event-based SLA violation detection. In: Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing -MWSOC 2009, pp. 1–6. ACM Press, New York (2009), doi:10.1145/1657755.1657756
5. Roxburgh, D., Spaven, D., Gallen, C.: Monitoring as an SLA-oriented consumable service for SaaS assurance: A prototype. In: Proc. Integrated Network Management, pp. 925–939 (2011)
6. Wang, G., Wang, C., Chen, A., Wang, H., Fung, C., Uczekaj, S., Chen, Y.-L., Guthmiller, W.G., Lee, J.: Service Level Management using QoS Monitoring, Diagnostics, and Adaptation for Networked Enterprise Systems. In: Proceedings of the Ninth IEEE International EDOC Enterprise Computing Conference, pp. 239–250. IEEE Computer Society (2005)
7. Wang, Q., Liu, Y., Li, M., Mei, H.: An Online Monitoring Approach for Web services. In: 31st Annual International Computer Software and Applications Conference (COMPSAC 2007) (Compsac), vol. 1, pp. 335–342. IEEE (2007), doi:10.1109/COMPSAC.2007.70

Simulation-Based Performance Evaluation of Cloud Applications

Antonio Cuomo, Massimiliano Rak, and Umberto Villano

Abstract. As the cloud paradigm gains widespread adoption, performance evaluation and prediction of applications remain daunting tasks that have not yet been fully accomplished. Nevertheless, reliable performance numbers are the key to take the cloud to the next step, in which it will be possible to predict the maintenance cost of the applications and to introduce richer Service Level Agreement between the service provider and consumer. In this paper we propose a methodology based on both benchmarking and simulation which aims at predicting the performance of cloud applications developed trough the mOSAIC framework. The methodology is general enough to be of interest to other component-based cloud application development platforms which are emerging in these days.

1 Introduction

The emerging Cloud Computing paradigm founds on a pay-per-use model where all the resource provisioning is delegated to the *network* (behind the scene, one or more remote cloud providers). Resources are acquired only when actually needed and charged on the basis of their effective usage. In addition, as described in the NIST [5] definition, resources are offered in a *self-service* manner, in which no human involvement is needed other than the request from the service user. As the cloud paradigm grows in maturity, upper layers of abstractions are appearing in research and enterprise cloud offerings which provide new building blocks to ease the development of *cloud applications*. Founding on higher level abstractions like messaging and data storage systems, these applications are designed independently from the specific provider, and then executed on platforms deployed on the resources

Antonio Cuomo · Umberto Villano
University of Sannio
e-mail: {antonio.cuomo,villano}@unisannio.it

Massimiliano Rak
Second University of Naples
e-mail: massimiliano.rak@unina2.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 263–269.
springerlink.com

acquired by different cloud providers. Such applications are able in a (semi) automated way to acquire new cloud resources (i.e. components and virtual machines) as the service demand grows. The mOSAIC project [7] is one of the first examples of this approach to cloud application development.

While it is well known that cloud-based services tend to exhibit good scalability as the number of user requests grows and more resources are elastically acquired, very few results are at the state of art available on the prediction and evaluation of cloud application performance. In our opinion, this is a research problem worth of further investigation, as a reliable prediction of performance in the cloud will enable a more thorough evaluation of business risks (by predicting application maintenance costs) and the introduction of richer Service Level Agreements for the negotiation of the quality of service between the provider and the consumer.

In this paper we propose a simple methodology to obtain performance prediction of mOSAIC-based cloud applications. The proposed approach is based on the following idea: for a target application, it is possible to build up a set of custom benchmarks to drive a simulation model that is able to predict the performance of the application on the set of available resources. The mOSAIC benchmarking framework helps in building custom benchmarking applications, which can be run on the target resources *on-the-fly*. The numbers obtained in the benchmarks are used to determine the parameters of simulation models of the cloud components: these models are described through JADES, a Java-based library which allows to specify and evaluate process-oriented discrete-event simulations.

The main drawback of such an approach is, obviously, that it is resource-consuming (so it affects the overall application resource usage). On the plus side, this operation can be done at application startup or when the application is not under stress, and offers clear optimization of the resource usage. A main goal of the methodology will be to limit the resource consumption of the simulation and benchmarks execution.

The remainder of the paper is structured as follows. Section 2 describes the structure of mOSAIC applications and the proposed performance evaluation methodology. The following section 3 shows how to apply the proposed methodology with a simple case study. The paper ends with a section dedicated to conclusions and plans for future work.

2 Performance Evaluation of mOSAIC Applications

In mOSAIC, a cloud application is structured as a set of interconnected components running on resources (i.e. Virtual Machines) leased by a cloud provider. The mOSAIC Platform, which enables mOSAIC application execution, manages the resources in a transparent way. Pre-defined mOSAIC components ease the development of applications: **Queuing systems** (*RabbitMQ* and *zeroMQ*) are used for component communications, an **HTTP gateway** accepts HTTP requests and forwards them to application queues, and **NoSQL data management systems** like Key-Value (KV) stores and columnar databases are used to organize application data.

The mOSAIC Java-based API enables the development of application-specific components in the form of *cloudlets*: these are stateless, event-driven and asynchronous components that can self-scale on the above described platform and consume any kind of cloud resource such as queues, independently of the technologies and the API they adopt, through a wrapping system [1, 6].

A mOSAIC cloud application is described as a whole in an *Application Descriptor*, which lists all the application components, the cloud resources (i.e. queues and key-value stores) and the details of their interconnections. Using the descriptor as a recipe for instantiating the components on the mOSAIC platform, the resulting application can then be provided in the form of Software-as-a-Service, to be accessed by final users.

Moving from these premises, we propose a methodology to generate directly from the application descriptor a performance model of the application that can be used to predict its performance, to tune its execution and to form the basis for Service level Agreement negotiation. The methodology can be schematized in the following steps: (1) Derive the simulation model from the Application Descriptor, (2) Derive the benchmarking applications from the Application Descriptor, (3) Execute the benchmark and collect the results, (4) Execute the simulation using benchmark results as parameters.

Simulation generation. In the first step, simulation models are derived directly from the application descriptor, by substituting the mOSAIC components with equivalent simulation models. The simulation models are organized in: the logical application, a set of simulation processes which mimic the behavior of each component in the descriptor; the resource models, that capture the behavior of execution components like the virtual machine CPUs, memory and storage subsystems; a mapping function which associates the application components with the resources on which they execute.

Benchmark generation. For the second step, mOSAIC offers a benchmarking framework which helps the mOSAIC user to build up custom benchmarks targeted to specific performance goals. Through the framework, it is possible to derive dedicated benchmarking application for each of the application component in the descriptor.

Benchmarks in the cloud. A notable feature is that the benchmarks are themselves mOSAIC applications that can be run on the target resources. The framework includes workload generators, components which help in controlling benchmark execution and components dedicated to gather benchmarking results.

Simulation in the cloud. The simulation models are also mOSAIC applications. In recent work, our group proposed mJADES [2, 3], a system able to produce simulation tasks from simulation jobs and to run them in parallel by means of multiple instances of the simulation engine, which are executed as mOSAIC cloudlets. The output variables or performance indicators are successively aggregated as needed, and presented to the final user. The mJADES application uses the JADES (JAva Discrete Event Simulator) library described in [4] to produce the simulation models.

3 A Case Study: XML Document Analyzer

In order to show how to apply the proposed methodology we focus on a simple case study: a mOSAIC application able to analyze XML documents. Such application is really simple in its architecture, as shown in Figure 1.

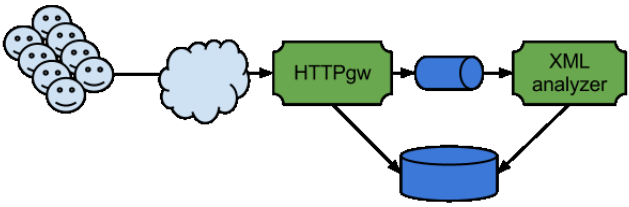


Fig. 1 the XML Analyzer mOSAIC Application

In order to predict the performance of the application we need to model and benchmark its resources (Queue and Key-Value store) and components (XML analyzer and HTTPgw).

An example benchmark is shown in Figure 2: the Queue component is tested by submitting a fixed number of workload messages and evaluating the average response time and resource usage of the Queue. Due to space limitations we do not report here the similar benchmarking applications for the other components.

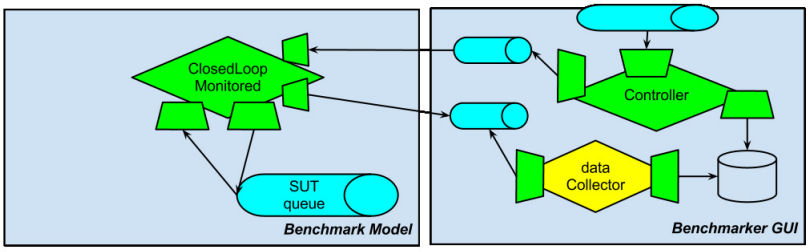


Fig. 2 The Queue benchmark mOSAIC Application

The simulation models are shown in Listing 4.1. Each component is modeled by a JADES simulation process which executes a potentially infinite loop: in practice, simulation termination is defined in the simulation driver routine (not shown here for brevity's sake), and can be linked to a specific duration or to the occurrence of an event. The components are interconnected through *mailboxes*, JADES components which allow for the modeling of interprocess communication. The `HTTPgw` component model act as an interface for mOSAIC application towards the external world. In the simulation, this component is the source of the external requests submitted to

the application, i.e. the simulation workload. The workload is characterized through the `waitTime` parameter and the `generateMessage` function. The former is used to model the interarrival times of user requests, while the latter can be customized to define the message characteristics. The process resource usage depends on the message and is modeled through the `Evaluate` function described next.

Listing 4.1 XMLAnalyzer Simulation model

```
void HTTPgwProcess(){
while (true) {
    hold(random.exponential(waitTime));
    Message m = generateMessage();
    mboxQueueIn.send(m);
    mapping.getResource().consume(Evaluate(m));
}}

void queueProcess(){
while(true) {
    Message m=mboxQueueIn.receive();
    mapping.getResource().consume(Evaluate(state,m));
    mboxQueueOut.send(m);
}}

void XMLAnalyzerProcess{
while (simulation=active) {
    m=mboxQueueOut.wait()
    mapping.getResource().consume(m);
    mboxKvIn.send(m);
}}

void KVProcess(){
while (true) {
    Message m = mboxKvIn.wait();
    mapping.getResource().consume(Evaluate(state,m));
}}
```

The `Queue` process executes a loop in which it receives incoming messages, consumes computational resources to account for queue processing and forwards the messages on the outgoing mailbox.

The next component is the `XMLAnalyzer`. Differently from the other components, which are natively provided by the mOSAIC framework, this one is a cloudlet developed trough the mOSAIC API, so only the mOSAIC developer knows exactly its behaviour. Nevertheless, the detail of its interconnections in the descriptor are sufficient to generate the simulation model: the cloudlet picks up messages from the queue, perform its resource-consuming elaboration and sends the results to the KV store. Finer-grained interactions (e.g. drawing two messages for each elaboration)

can be supported by manually modifying the model. The missing step is the construction of a custom benchmarking application, similar to the one adopted for the Queue.

The final process represents the `KV` component, which as shown in the code receives incoming storage messages and consumes resources when storing data. For each component of the simulation model, the benchmarking results will be used to fill the parameters of the `Evaluate` functions, which are regressive functions constructed on the benchmark figures.

4 Conclusions and Future Works

The evelopment of cloud applications is a complex task, in which resource optimization has a relevant role. In this paper we have proposed a simulation-based methodology which helps the developer to predict the performance of his application given the acquired resources and the requests of the end users. The approach here proposed founds on performance simulations tuned by benchmarking. The benchmarks are based on *ad-hoc* applications that can be easily replicated in order to take into account eventual variations on the resources acquired. The included case study describes the derivation of the simulation models and benchmarks for a simple application. In future work we will extend the approach in order to enable the application to self-tune itself on the basis of the end-user requests, autonomously starting both benchmarks and simulations to take decisions on the acquisition or release of the resources. In addition we aim at integrating such functionalities with the Service Level Agreement (SLA) Framework offered in mOSAIC, so that performance simulation and benchmarks can be used in order to negotiate the SLAs and predict eventual violations.

Acknowledgements. This research is partially supported by FP7-ICT-2009-5-256910 (mOSAIC)

References

1. Craciun, C.D., Neagul, M., Lazcanotegui, I., Rak, M., Petcu, D.: Building an Interoperability API for Sky Computing. In: The 2nd Int. InterCloud Workshop, pp. 405–406. IEEE (2011)
2. Cuomo, A., Rak, M., Villano, U.: Cloud-based Concurrent Simulation at Work: Fast Performance Prediction of Parallel Programs. In: 2012 21th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE (to appear, 2012)
3. Cuomo, A., Rak, M., Villano, U.: mJADES: Concurrent Simulation in the Cloud. In: 2nd Int. Workshop on Intelligent Computing at Large Scale (to appear, 2012)
4. Cuomo, A., Rak, M., Villano, U.: Process-oriented discrete-event simulation in Java with continuations: quantitative performance evaluation. In: Int. Conf. on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH (to appear, 2012)

5. Mell, P., Grance, T.: The NIST definition of cloud computing. NIST Special Publication 800, 145 (2011)
6. Petcu, D., Craciun, C., Rak, M.: Towards a Cross Platform Cloud API - Components for Cloud Federation. In: Leymann, F., Ivanov, I., van Sinderen, M., Shishkov, B. (eds.) CLOSER, pp. 166–169. SciTePress (2011)
7. mOSAIC: Open Source API and platform for multiple Clouds (2010), http://www.mosaic-cloud.eu

Vendor Agents for IAAS Cloud Interoperability

Alba Amato, Luca Tasquier, and Adrian Copie

Abstract. Elastic provisioning of Cloud resources at IAAS is a mandatory facility to deploy applications on computing elements which are dynamically allocated based on the application needs. Because of the increasing offer of the Cloud market, the effectiveness of provisioning can be increased by selecting and exploiting the best proposal that is compliant with the user's requirements. Due to the current lack of standards and to the heterogeneity of technologies interoperability at IAAS, it is the main issue to be addressed. We propose an agent based solution to abstract IAAS services for negotiation and management of Cloud resources. The paper presents our agent abstraction and its implementation to support two well known Cloud technologies: Open Nebula and Amazon.

1 Introduction

Cloud computing provides an alternative and more agile way of using IT services allowing a more rapid and elastic reaction to market conditions and giving the possibility to scale up and down as needed. It represents both a technology for using computing infrastructures in a more efficient way and a flexible, pay-as-you-go business model for selling computing resources. However to enjoy these benefits, it is necessary to select the provider whose offer best fits the user's needs by taking careful consideration in order to fully evaluate features and pricing of the cloud providers. In fact, the choice of a cloud computing vendor is really difficult because there are thousands of options and there are some key factors to consider in terms of characteristics of the service, general terms and conditions of service and service levels that

Alba Amato · Luca Tasquier
Department of Information Engineering, Second University of Naples
e-mail: alba.amato@unina2.it, luca.tasquier@gmail.com

Adrian Copie
Research Institute e-Austria, Timisoara, Romania (IeAT)
e-mail: adrian.copie@info.uvt.ro

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 271–280.
springerlink.com

providers ensure. Apart of that, the cloud providers offer heterogeneous resources and technology which are described according to their own metrics or properties which can change from one provider to another. To determine which provider meets the user's needs it is necessary to evaluate the list of services the provider will deliver against a complete definition of each service, the cost per unit of capacity for each provider, the mechanism and the metrics provided to monitor the service and to determine whether the provider is delivering the service as promised, the responsibilities of the provider and of the consumer and remedies available to both if the terms of the SLA are breached, the security designations used by the providers, etc. In order to address the problem of the heterogeneity of the offers it is also necessary to consider that the descriptions of the offers must be understood by the users and compared to their requirements in order to make an efficient choice taking into account not only the technical requirements but also the providers rules and conditions. This paper addresses the problem of provisioning and management of resources at Cloud infrastructure level (IAAS) with particular attention to problems of interoperability and portability arising from the heterogeneity of technology whose solution is one of the main objective of mOSAIC project. The mOSAIC project (EU FP7-ICT programme) [11] intends to progress the state-of-the-art in Cloud computing by creating, promoting and exploiting an open-source Cloud application programming interface and platform targeted for developing multi-Cloud oriented applications and to negotiate Cloud services according to application requirements. The main benefit of using the mOSAIC software package will be a transparent and simple access to heterogeneous Cloud computing resources and avoidance of lock-in proprietary solutions. Having this aim, we propose an agent based solution to abstract IAAS services for negotiation and management of Cloud resources that actively perform its functions within the Cloud Agency, a MAS that have the main task of dynamically selecting a set of Cloud resources, from different vendors, that best fit the users requirements. Cloud Agency [15] complements the common management functionalities which are currently provided by Infrastructure as a Service (IAAS) Private and Public infrastructure with new advanced services, by implementing transparent layer to IAAS Private and Public Cloud services.

Here we present our agent abstraction and its implementation to support two well known Cloud technologies: Open Nebula and Amazon.

2 Related Works

The demands for standards in clouds to ensure portability, interoperability and integration of Cloud providers is a complex issue due to the heterogeneity of cloud platforms and providers. During last years, there were some efforts in the open-source community aimed on solving the problem of the standardization between different cloud service providers. Obtaining a common interface is the focus of the OCCI group that defined a model for Cloud management at IAAS. OCCI defines entities, API and protocol. The specification of Cloud API is a Resource Oriented Architecture (ROA) that uses REpresentational State Transfer (REST)protocol. The OCCI

core meta-model [12] provides means of handling abstract Cloud resources. libcloud [3], an Apache incubator project, is a client library for interacting with many of the popular cloud server providers by providing a uniform API which translates API calls to the formats proprietary cloud APIs can understand. jclouds [4] is an open source library that offers several API abstractions as java and clojure librarie supporting of 30 cloud providers and cloud software stacks. The Simple Cloud API [7] is a common API for accessing cloud application services offered by multiple vendors that supports four cloud application services: File Storage Services, Document Storage Services, Queue Services, Infrastructure. Red Hat's Deltacloud [2] project is developing a open source standardized API for addressing different cloud architectures in a uniform way. However, the above projects do not address the mechanisms of negotiation, monitoring and provisioning. SLA@SOI [8] is the main project that aims (together with other relevant goals) at offering an open source based SLA management framework. It will provide benefits of predictability, transparency and automation in an arbitrary service-oriented infrastructure, being compliant with the OCCI standard. SLA@SOI results are extremely interesting and offer a clear starting basis for the SLA provisioning and management in complex architectures. SLA@SOI offer solutions to design Cloud services with multi-level and multi-provider SLAs. The main target of the project is Cloud Provider enrichment for offering a solution that can be integrated with Cloud technologies (like Open Nebula) trough an SLA-based approach. It defines a syntax for machine-readable Service Level Agreements (SLAs) and SLA templates. Machine-readable SLAs, will allow consumers and providers of online services to precisely specify the services and service levels they require,confirm that SLAs are being met, and automatically deal with any SLA violations allowing service levels to be personalized, automatically negotiated, aggregated, and continuously assured [16]. In the above projects, API are designed either to interface only with programming languages like Java, Python or PHP, or they are providing connectors or wrappers to a small number of provider offers. There is no cross platform API that is provider and language independent, and also aims the auto-scalability of applications [14]. Within the mOSAIC project, we designed and implemented a reference set of API which intends to be a language independent and programing paradigm free. Besides Cloud Agency implements a multi-agent brokering mechanism of provisioning, monitoring and management that is transparent with respect to the provider. In particular, the vendor agent abstracts the functionalities of provisioning and management. The above characteristics differentiate the mOSAIC's API from the presented general API proposals for unified Cloud resources representations that mainly wrap the native APIs of the hosted Cloud services seen before [13].

3 The Vendor Agent for mOSAIC

There are two important business processes which are relevant in relation with the vendor agents: *the resource provisioning* and *the resource management*. The resource provisioning activities are intended to provide an appropriate vendor

proposal for a Call for Proposal (CFP) received by the agent. Eventually, such a proposal becomes an agreed proposal: Service Level Agreement (SLA). The operations performed by the vendor agent in this regard include CFPs/SLAs management and proposal generation. The cloud providers offers a minimal support related to the providing process. Most of the providers do not provide such information in a convenient form for automatic processing. Therefore, the vendor agents should be kept up to date in relation with the cloud providers offers by either automatically parsing the providers web pages or importing services description after manual user processing. The resource management activities aim to allow operations such as the creation or the deletion of cloud resources but also to perform a set of operations on the created resources. The management of the resources is related to the result of the provisioning phase which can be described into a special kind of a descriptor which bounds the provisioned cloud resources to cloud providers and SLAs. The structure of a vendor agent is also affected by the agent infrastructure it is plugged in. We consider JADE as the infrastructure provider for our agents. Therefore the agents should include a layer to ensure the integration with the JADE agency middleware. The figure 1 presents the overall view of the vendor agents as they integrate in the agency. The agents have to deal with provider specific elements and it is the *Vendor*

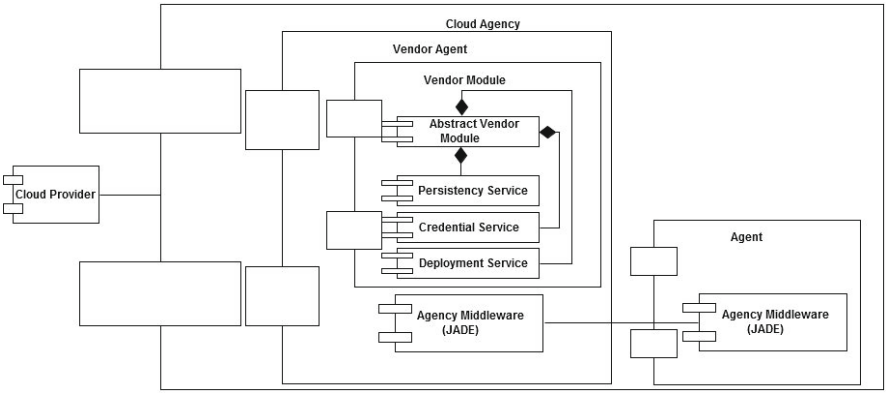


Fig. 1 The Vendor Agent

Module (VMod) component which is intended to cover the cloud provider specifics. The VMod interacts with the provider in order to perform resource provisioning and management. It uses specific libraries and protocols in order to authenticate with the provider, to create resources of specific types and to perform operations on them. This module also generates proposals for CFPs based on the specifics of the provider. The *Vendor Module* extends the functionality of an *Abstract Vendor Module* (AVMod) which is intended to implement the common aspects on each provider. Such aspects include persistence of proposals data and resource information, credentials management or the deployment descriptions. The AVMod is also responsible for the integration of the agent with the other agents. It receives requests

from other agents, delegates the servicing to the VMod and provide callbacks for the VMod in order to allow the other agents perceiving the results of the requests. The resource provisioning operations provided at the level of the AVMod are:

- *submitCFP*: accepts a CFP and delegates to the VMod the servicing;
- *onProposalAvailable*: callback for VMod in order to signal the availability of a proposal;
- *onCFPReject*: callback for VMod to signal CFP rejection because of not being able to create a proposal;
- *deleteProposal*: the client specifies that a proposal is no longer interesting and should be discarded;
- *acceptProposal*: the client accepts a proposal and thus become an SLA. It also provides a deployment description which allows the agent to manage the proposal's resources;
- *refuseProposal*: the client refuses a specific proposal.

The operations related to the resource management includes requests coming from the clients which will be implemented at the level of VMods:

- *createResource*: creates a resource of a specified class for the specified SLA. Resources can be created at any moment;
- *releaseResource*: releases a previously created resource;
- *performAction*: performs a specified action on a resource.

There are also some callbacks being provided for the VMods:

- *onResourceCreation*: signals the client that a resource was created and information about it is available;
- *onResourceRelease*: confirms the resource release to the client;
- *onResourceActionPerformed*: confirms an action on a resource;
- *onResourceStatusUpdate*: updates the client about one resource's status.

Complementary callbacks for perceiving failures are also available. The AVMod is informed by the credentials data coming from clients through the *onCredentials* operation.

4 A Comparison of Amazon EC2 and OpenNebula

Amazon Amazon Web Services [1] is a collection of web services aggregated into a cloud computing platform that provides a wide range of resources like *computation*, *storage*, *network*, *communication* and *database* that can be orchestrated and used in order to develop extremely complex applications that benefit of its proven hosting and execution platform in the public cloud. The computational resources are offered through the Elastic Compute Cloud (EC2) [1] platform which provides resizeable compute capacity in the cloud and exposes a true virtual computing environment able to launch and manage instances of various operating systems. The EC2's resources are images of virtual machines (VMs) that can be used off the shelf or can be previously uploaded in the Amazon cloud and then managed in the same manner

as the predefined images. Amazon offers a wide range of predefined images with different configurations and containing various operating systems. For special needs, customized images could be locally prepared and uploaded in the cloud in order to be instantiated and executed. The customized images management is done through a series of utilities programs instead of API calls. The operations performed over the Amazon images are creation, upload in the cloud, download from the cloud and delete. The instances are completely controlled due to the provided root access and they can be managed as any local machine. The flexibility in choosing the operating system, the configuration in terms of CPU, RAM or storage make possible to select the optimal environment for any application. The VM instances support operations like *run*, *stop*, *start*, and *terminate*, which describe an entire lifecycle for an instance. They can be accessed either via REST or SOAP protocols, but also a Java API is provided, which basically relies on REST calls but offers a much friendly development experience for programmers. An EC2 instance can be launched with a choice of two types of storage for its boot disk: the local instance storage which does not persist the data after the instance is terminated, and an Elastic Block Store (EBS) which provides persistence independent of the lifecycle of an EC2 instance. The Amazon platform provides numerous security mechanisms that protect the computing resources like services through firewall settings or establishing IP ranges used to facilitate the connection via industry standard encrypted IPSec. At a higher level, Amazon offers the Amazon Identity and Access Management (IAM) service which can track the users identities and grant granular permissions on different resources. Amazon does not provide a negotiation mechanism or API for the QoS, the same SLAs are available for all the customers.

Open Nebula. OpenNebula [6] is an open-source cloud computing toolkit for managing heterogeneous distributed data center infrastructures. The OpenNebula toolkit manages a data center's virtual infrastructure to build private, public and hybrid IaaS (Infrastructure as a Service) clouds. OpenNebula orchestrates storage, network, virtualization, monitoring, and security technologies to deploy multi-tier services (e.g. compute clusters) as virtual machines on distributed infrastructures, combining both data center resources and remote cloud resources, according to allocation policies. The kind of VM images that can be run generally depends on the hypervisors installed on the hosts and on the OpenNebula version. Using the 3.2.1 release, OpenNebula can handle images of Xen [10], KVM [5] and VMware [9]. It can manage pratically all the cloud services available in IaaS, such as VM (with and without mounted image), storage, image and network. Each resource can be accessed in several ways: REST (OCCI compliant), EC2 compliant API, XML-RPC API, Ruby API, Java API. The available operations depends on the particular managed resource. For a VM, it's possible to perform the common management operations, such as start, stop, reboot, suspend and resume, and some specific ones like hold, release, finalize and cancel. For an Image, it's possible to upload, publish, unpublish, allocate, enable, disable and delete the resource. For a Network the available operations are quiet the same that can be done for an Image (allocate, publish, unpublish, etc.). By default most of the interfaces to OpenNebula communicate with

the core by using XML-RPC calls that contain the user's session string, which is authenticated by the OpenNebula core by comparing the username and the password against the registered users. Optionally OpenNebula can delegate the authentication to external modules. It is supported the CLI authentication with SSH authentication and X.509 but also Server authentication where by default the forwarded requests are encrypted by using a simmetric key mechanism.

The Instances types offered by OpenNebula depends on provider that it uses while Amazon provides a fixed number of Instance Types. Also about negotiation's mechanisms, OpenNebula doesn't provide any API or system to integrate prices and/or QoS negotiation, so they depend on the providers that use OpenNebula.

Table 1 Summary of Amazon and OpenNebula

	Amazon	OpenNebula
Offered Resources	Computation, Storage, Network, Communication, Database	Storage, Network, Computation
Image Supported Operations	Bundle, Upload, Download, Delete	Upload, Publish, Unpublish, Allocate, Enable, Disable, Delete
VM Supported Operations	Run, Stop, Start, Reboot, Terminate	Start, Stop, Suspend, Reboot, Resume, Hold, Release
Access VMs	REST, SOAP, Java API	REST (OCCI compliant), EC2 compliant API, XML-RPC API, Ruby API, Java API
Security	IP ranges, Identity and Access Management	XML-RPC or by delegating to external modules (SSH, X.509)
SLA Negotiation	No	No
Instance type	Fixed Number	Depends on provider

In summary, both Amazon and OpenNebula are powerful solutions for building up flexible computation infrastructure. While Amazon aims to offer practically unlimited resources and extremely reliable SLAs, targeting a wide range of users, OpenNebula is geared mostly for the small companies that want to quickly build a private cloud computing environment and use it internally. Table 1 summarizes the most important characteristics of both cloud infrastructure systems.

5 Vendor Agent for Open Nebula

The Vendor Agent for OpenNebula uses the Abstract Vendor Module in order to communicate with the Cloud Agency. For the development process are being used the Java API. This choice have two main reasons: the former is that the Cloud Agency is written in Java and so there is the full compatibility with all the agents infrastructure. The second reason is that the Java API is a wrapper of the XML-RPC: all the OpenNebula installations start by default the XML-RPC server, so the communication between the Vendor Module and the OpenNebula core can occur without

any other management operation. For the RPC calls, the module uses an HTTP address that represents the OpenNebula core's location. This endpoint is loaded by the Vendor Module at boot time. The first overridden method is *submitCfp*: the module receives the Call For Proposal and the generated transaction ID from the Cloud Agency. Then it translates the received document in a OpenNebula compliant format and queries the OpenNebula core to retrieve the possible offers. After that it compares the received offers against the Call For Proposal and chooses the one that best fits with it. Currently the choosing policies are based on the order of the parameters described in the Call For Proposal: for instance, if is described a VM with 512 MB of memory amount as the first parameter and 1 GHz of cpu speed as the second one, the VMod compares its offers against the memory amount at first and then compares the cpu speed, taking in account the fitting of the memory amount with the requested one as primary criterion. At the end of this provisioning step it translates the chosen offer to an agnostic format and calls the *proposalAvailable* method to send to Cloud Agency the offer. It's very important to focus the attention on the actions executed when an offer is accepted. The Vendor Module overrides the *acceptProposal* method retrieving from it three informations: the accepted offer ID; the proposal; the user's credential to access the OpenNebula account.

First of all, the Vendor Module checks the user's credentials. If these ones are not provided by the user or the authentication fails, the Vendor Module sends a message to the Cloud Agency requesting for the authentication's parameters. When the authentication succeeds, the module connects to the XML-RPC server by using the endpoint's address and the received credentials. After the connection's step, the Vendor Module loads the template that matches with the chosen offer.

In a OpenNebula template are stored some important informations about the selected resource: for example to load a VM the template can store informations about the percentage of physical CPU reserved to the VM, the amount of memory, the attached image's description (it's possible to upload and attach the image in a second time), the attached disk's description and the used network configuration. After choosing the right template, the Vendor Module try to instantiate the selected resource. If the allocation succeeds, the module returns the resource's ID to the Cloud Agency. The described operations are the main ones of the provisioning functionalities. Regarding the management operations, it's possible to analyse the overridden *performAction* method: as the *acceptProposal* functionality, the first step consists in connecting to the OpenNebula core by using the user's credentials via the vendor's endpoint. Once the connection has been established, the module recovers the selected resource by using the resource's ID. After that it uses the passed action and parameters to translate the agnostic request into the operations which lead to the compliance with the request. If the action succeeds, the Vendor Module notifies this event to the Cloud Agency via *performed action* method. If the operation isn't supported or the request is bad-formed, the module notifies a message to the Cloud Agency by using the *performAction Failure* method.The currently supported actions are *start*, *stop*, *reboot* and *destroy*.

6 Vendor Agent for Amazon

The vendor agent that deals with the resources provided by Amazon Web Services is described in a class called AmazonVendorModule, and it is implemented in Java, as the Open Nebula vendor agent component. The reason behind this decision is to tight the Amazon implementation to the existing libraries and APIs on the market, many of them being written in Java. In order to connect to Amazon and perform the resource provisioning and resource management, the *typica* library was used, which transforms the Java calls to REST calls in the Amazon Web Services. Basically the Amazon Vendor performs the same operations described in section 3 but also implements some operations that are specific to Amazon. These particular operations are isolated in the *doCreateResource* method for resources creation, *doReleaseResource* for resources destruction and *performAction* for the resources management operations. The specific operations supported in performAction method are *stop*, *start* and *reboot* the virtual machines. In the same time the Amazon Vendor Module supports operations related to securely upload files on the computation resources and also related to remote programs execution on the remote instances making possible the placement of a list of files on it and execution of various programs remotely. The Amazon credentials are managed outside the vendor agent, they are not stored within the Java components and they are required directly from the application for security reasons. Table 2 summarizes the differences between Amazon and OpenNebula Vendor Modules.

Table 2 Differences between Amazon and OpenNebula Vendor Modules

	Amazon	OpenNebula
Currently Supported Actions	Create, Start, Stop, Reboot	Create, Start, Stop, Reboot, Destroy
Used Libraries	Typica	Java OCA
Credentials Management	Outside Vendor Agent	Inside Vendor Agent
Credentials Implementation	A tuple access key: secret key is used to access the Amazon resources and to perform various actions on them	A tuple username: password is used and for each operation credentials are needed
Wrapper Used	REST Calls	RPC-XML Calls

7 Conclusions

In this paper we focused on the problem of provisioning and management of resources at Cloud infrastructure level (IAAS) with particular attention to problems of interoperability and portability arising from the heterogeneity of technologies. To address this problem, we presented an agent based solution integrated within the Cloud Agency and its implementation for supporting Open Nebula and Amazon cloud solutions. The vendor agent represents a clear innovation in its ability to abstract the functionalities of provisioning and management and in its proactivity

in offering the proposal and participating in the negotiation, showing that it can be possible to let two cloud frameworks interoperate even with their different architectures, with the overall advantage that cloud service can be deployed and managed in a environment which best fits their needs. The further evolution of this solution that constitutes an ongoing activity is the implementation of the vendor agent to supports other providers.

Acknowledgements. This work has been supported by the mOSAIC project (EU FP7-ICT programme, project under grant #256910) and by PRIST 2009, Fruizione assistita e context aware di siti archelogici complessi mediante terminali mobile, funded by Second University of Naples. The authors would like to thank Antonio Bagarolo for his technical support.

References

1. Amazon: (2012), http://aws.amazon.com/
2. Apache deltacloud (2012), http://deltacloud.apache.org/
3. Apache libcloud (2012), http://libcloud.apache.org/
4. jclouds (2012), http://www.jclouds.org/
5. Kvm (2012), http://www.linux-kvm.org/
6. Opennebula (2012), http://www.opennebula.org/
7. Simple cloud api (2012), http://simplecloud.org/
8. Sla@soi (2012), http://sla-at-soi.eu/
9. Vmware (2012), http://www.vmware.com/
10. Xen (2012), http://www.xen.org/
11. DiMartino, B., et al.: Building a mosaic of clouds. In: Proceedings of the 2010 Conference on Parallel Processing, Euro-Par 2010, pp. 571–578. Springer (2011)
12. Metsch, T., et al.: Open cloud computing interface – core and models. In: Standards Track, Muncie (IN). GFD-R, The Open Grid Forum Document (2011)
13. Petcu, D.: Portability and Interoperability between Clouds: Challenges and Case Study. In: Abramowicz, W., Llorente, I.M., Surridge, M., Zisman, A., Vayssière, J. (eds.) ServiceWave 2011. LNCS, vol. 6994, pp. 62–74. Springer, Heidelberg (2011)
14. Petcu, D., et al.: Building an interoperability api for sky computing. In: Proceedings of the International Conference on High Performance Computing and Simulation (HPCS), pp. 405–411. IEEE (2011)
15. Aversa, R., Di Martino, B., Venticinque, S.: Distributed agents network for ubiquitous monitoring and services exploitation. In: 12th IEEE International Conference on Computational Science and Engineering, pp. 197–204 (2009)
16. Wieder, P., et al.: Service Level Agreements for Cloud Computing. Springer (2011)

Agents Layer to Support Cloud Applications

Calin Sandru and Salvatore Venticinque

Abstract. The process of developing, deploying and executing cloud applications is greatly influenced by the specifics of the cloud providers regarding the cloud infrastructure and the cloud resources. Important challenges are related to agreeing with the cloud vendors about the application resources and the quality of the services. Migrating the application from one cloud provider to another cloud provider or even using multiple providers at once is also difficult to achieve. The present paper proposes an architectural solution for the above mentioned problems by considering the agency paradigm and a special set of agents called *Vendor Agents* abstracting the cloud provider differences.

1 Introduction

Cloud Computing has become today a real opportunity for provisioning of Computing resources from the network. Unfortunately two main issues affect both the possibility of choosing in a comfortable way the best Cloud solution/offer and of migrating Cloud applications. In fact, the first problem deals with the big gap that exists between two Cloud offers. The heterogeneity of Cloud services regards not only the provided facilities, but also the terms of services, the guarantees, the service levels and the metrics by which they are measured. Last but not least a common ontology, for disambiguating the semantic of the offers to be evaluated still lacks and make this an hard task also for an expert user. About the heterogeneity of available Cloud offers, every big commercial provider is interested in proposing a proprietary solution to lock its customers. Also open source technologies for setting up private

Calin Sandru
Research Institute e-Austria, Timisoara, Romania (IeAT)
e-mail: csandru@info.uvt.ro

Salvatore Venticinque
Second University of Naples, Aversa, Italy
e-mail: salvatore.venticinque@unina2.it

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 281–286.
springerlink.com

Clouds are not compliant to each other, because we still miss a wide accepted standard. It means that any choice affects the interoperability of the developed Cloud applications with other technologies, and would compromise the convenience of a better business, due to the considerable effort for re-engineering the developed software. The work presented in [3] represents a first proposal to combine SLA-based resource negotiations with virtualized resources in terms of on-demand service provisioning. Cloud interoperability is the main issue here. In particular OCCI (Open Cloud Computing Interface) is a proposal of a standard in Cloud [4]. It defines entities, relationships API and protocols for all kinds of management tasks at IaaS. As in our approach, extension is used to improve interoperability but, in this case, it does not add new functionalities. Rather than adopting a standard or developing a new service interface, there exists on-going efforts in order to ensure a common API in the cloud. DeltaCloud aims to provide access to compute resources and to different storage options (volumes and buckets) using a REST API. Libcloud and JClouds are other in the field which provides access by language specific API to compute and storage resources on a large set of cloud providers. The exploitation of agents technology in this context is investigated in [2]. A simpler agents based architecture has been proposed in [6]. Preliminary investigations by the authors on related topics were presented in [5]. In this paper we address both the problems of provisioning and management of resources at Cloud infrastructure level (IAAS). We designed a Vendor Agent to abstract the services of different IAAS Cloud vendors. This abstraction is used to provide a uniform communication model among agents that implement value added services within a more complex framework: Cloud Agency. Cloud Agency is a multi-agents system that has designed and developed by the research activities of the European Project mOSAIC[1], for provisioning, management and monitoring of Cloud resources. Vendor Agents allow the agency for inter-operating with different Cloud Providers. Furthermore they do not only wrap different technologies, but provide intelligence for supporting negotiation and management of Cloud resources.

2 Cloud Agency

Cloud Agency is a multi agent system (MAS) that accesses, on behalf of the user, the utility market of Cloud computing to always maintain the best resources configuration that satisfies the application requirements. This system is being designed and developed within the research activities of the FP7 mOSAIC project. It is in charge to provide the collection of Cloud resources from different vendors that meets the requirements of the users' applications. According to the available offers, it generates a Service Level Agreement that represents the result of resource brokering and booking with available providers. The user is able to delegate to the Agency the monitoring of resource utilization, the necessary checks of the agreement fulfillment and potential re-negotiations. Cloud Agency will supplement the common management functions currently provided by IAAS Private and Public

[1] www.mosaic-cloud.eu

infrastructure with new advanced services by implementing a transparent layer of services for IAAS Private and Public Clouds. An important goal of the CA we are focusing here is to facilitate the application deployment process. This process can be regarded as a two-phase process: one phase which happens on the user computer and one which continues on the cloud. The first stage involves the resource provisioning. The user and the cloud provider(s) agree on the resources to use and the quality of the services. The CA is running on the user's computer as cloud resources are not yet provisioned. The second stage starts with the CA agents being restarted on the cloud. This way, the agents are close enough to the created resources in order to be able to perform other functions like resource management, monitoring and auto-scaling. At this point, the local agency is no longer needed and the user directly interacts with the cloud deployed agency. The different components of the

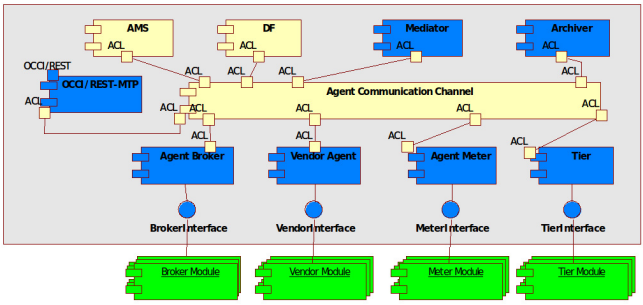


Fig. 1 Cloud Agency architecture

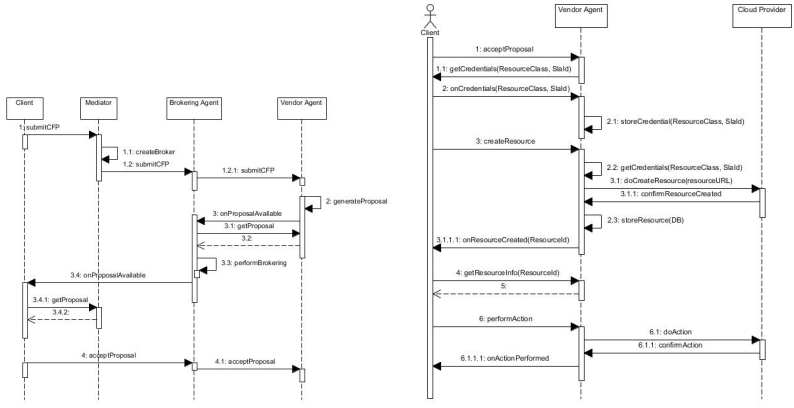
agency architecture are shown in Figure 1. It is important to note that the provided services are vendor independent, platform neutral and can be extended in order to solve a variety of problems in cloud computing. Our solution has been conceived as a proposal of extension of the OCCI to support provisioning, monitoring and reconfiguration. The Message Transfer Protocol (MTP) implements the OCCI HTTP rendering that is vendor independent and allows any client to invoke the services in an uniform way. Services are implemented by agents. RestFull messages are translated into ACL (Agent Communication Languages) and forwarded to the right agent receiver. Agents will communicate among them via standard ACL (Agent Communication Language) over HTTP, or over other transport protocols if necessary. Execution environment for agents and communication facilities are provided by the Jade agent platform[1]. Jade has been chosen as a platform to provide an execution environment of software agents, an Agent Communication Channel (ACC) and some protocol implementation to support communication. AMS and DF provide standard services of FIPA compliant agent platforms: a name server for agents and a yellow pages registry for publication and discovery of agent base services. The agent Mediator receives requests for starting new negotiations and retrieving information; the agent Broker gets multiple proposals from Vendor Agents and chooses the best one;

Meter and Archiver implement the monitoring service; the Tier is responsible to apply autonomic reconfiguration; the Vendor agent performs all the interactions with the Cloud provider. For the purpose of this paper we are mostly interested on the CA contributions to support the application deployment, specifically the resource provisioning and the cloud resource management. The agents contributing to the application deployment are the *Mediator Agent*, the *Broker Agents* and the *Vendor Agents*. The *Vendor Agents* are the core part of the management service as they receive client requests and mediates the interactions with the Cloud Providers.

3 Provisioning and Management across Heterogeneous Cloud Vendors

The overall goal of the Vendor Agents (VA) inside the Cloud Agency is to mediate the relationship of their clients with the specific cloud providers they are connected to. The VAs create a layer of separation between the Cloud Agency and the Cloud Provider and insulates the user's applications and other agents from the details of the cloud provider, whose resources they use or the infrastructure they run on. Vendor Agents basically deal with these issues along two dimensions: *the resource provisioning* and *the resource management*.

The resource provisioning refers to the activities performed by the VAs in order to achieve a *Service Level Agreement* (SLA) needed in order for a user application to run on a cloud provider infrastructure or to use its resources. The Vendor Agents provide these services within the Cloud Agency by implementing the protocol described in the Figure 2 (a). The starting point of the provisioning activities is the



(a) The resource provisioning process (b) The resource management process

Fig. 2 Cloud Agency interaction diagrams

Call for Proposal (CfP). The CfP includes the needed characteristics of the cloud resources to be used by the user's application and the CA itself. It also includes descriptions related to the Quality of Services. The descriptions are usually made in terms of patterns and ranges. The VA then proposes one or more SLAs on behalf of the cloud provider. At the time a SLA is proposed, the desired characteristics in the CfP are bound to specific values which can be accepted or not by the clients. In general, the cloud providers do not provide an API to negotiate resources. Instead, the commercial ones maintain tables with description of resources and pricing. In this case the vendor agent has to be aware of the content of the tables and to be updated on changes. Some cloud providers still allows for retrieving those tables from specific URLs (e.g. Amazon), in which case the vendor agents may have the chance to automate the process.

The resource management refers to the activities the VAs perform in order to facilitate the applications and other agents to operate on the cloud resources. Operations falling in this area include the creation and deletion of the resources, starting, stopping, rebooting compute resources, attaching storage to compute resources, etc. The SLA is a key ingredient when building an agency related application descriptor to allow the VAs knowing how to deploy the CA application resources. The VAs offer resource management facilities to the agency by being an actor of the sequence diagram shown in the Figure 2(b) . Of course the agent technology and the inter agents interaction will be hidden to the Client, that invokes OCCI RESTfull requests by the Cloud Agency interface. Along both the dimensions VAs creates artifacts to be further referred like SLA or cloud resources. The details of these artifacts have to be stored in some persistent media for further usage. There are two areas of storage which we consider: proposal storage and resource information storage. Therefore the VAs include a persistence service which is responsible with the details of storing the data in appropriate files. Even if the agents functionality is adapted the cloud providers specifics, some of the agents' functionality is still the same regardless of the cloud provider. Such functions include those related to the agent integration in the cloud agency, persistence and credentials management. Consequently, the agents can be structured in such a way that the common functionality can be reused across the agents.

The application deployment is a necessary facility to configure the full set of Cloud resources. Vendor Agents will be in charge for configuring the resources they are responsible for. There are few elements which worth to be mentioned. **Resource class** refers to a resource class as agreed and defined in the SLA. Depending of the resource class type, each instance of objects of a specific resource class is created based on the information to be provided in the resource class description. Such an element is the **Resource location**, that is the `<url>` indication which refers to the image of the virtual machine to load in the case of compute resources. The *hardware profile* concept in DeltaCloud could be regarded as a particular case of a resource class. **Resources relationship** depends on the resources classes types. Resource instances can be related in several ways. For example, some cloud providers offer load balancing resources and the compute resources of the same type can be *load balanced* on specific protocols and ports. Other relations which involve resources

in different classes are also possible. For example a storage volume resource can be attached to instances of compute resources. A **Resource setup** section can be used to set-up some resources. For example, the compute resources may require certain files/programs being uploaded onto their space before being usable and some scripts to run in order to properly initialize them (Libcloud also includes this concept in the form of the so called *bootstrap* scripts).

4 Conclusions

We exploited the potential of the agent oriented architectures while trying to address in a flexible way a set of issues related to Cloud provisioning and management. We have shown how the agents can help to manage Cloud infrastructures across cloud providers by allowing a uniform access. We have chosen to rely on cloud specific API by a RESTfull style of communication that is quite appropriate for the resource management operations performed by VAs through Cloud Agency. The Vendor Agents also controlled the application deployment process. Therefore the VAs could be considered at a higher layer of abstraction and the mentioned technologies could be used in order to facilitate a part of the VAs interaction with the cloud providers. The advantage would be the ability of interacting with many cloud providers without big efforts.

Acknowledgements. This research is partially supported by FP7-ICT-2009-5-256910 (mOSAIC) and by the grant POSDRU 21/1.5/G/13798.

References

1. Bellifemine, F., Caire, G., Poggi, A., Rimassa, G.: Jade - a white paper, in exp in search of innovation. Special Issue on JADE TILAB Journal (2003)
2. Cao, B.-Q., Li, B., Xia, Q.-M.: A Service-Oriented Qos-Assured and Multi-Agent Cloud Computing Architecture. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) Cloud Computing. LNCS, vol. 5931, pp. 644–649. Springer, Heidelberg (2009)
3. Kertesz, A., Kecskemeti, G., Brandic, I.: An sla-based resource virtualization approach for on-demand service provision. In: 3rd International Workshop on Virtualization Technologies in Distributed Computing (2009)
4. Metsch, T., Edmonds, A., et al.: Open cloud computing interface - core and models. Standards Track. GFD-R in The Open Grid Forum Document (2011)
5. Venticinque, S., Aversa, R., Di Martino, B., Rak, M.: Cloud agency: A mobile agent based cloud system. In: CISIS 2010 - International Conference on Intelligent, Complex and Intensive Systems, pp. 132–137. IEEE Comuter Society (2010)
6. You, X., Xu, X., Wan, J., Yu, D.: Ras-m: Resource allocation strategy based on market mechanism in cloud computing. In: Fourth ChinaGrid Annual Conference (2009)

Policy-Based Instantiation of Norms in MAS

Andreea Urzică and Cristian Gratie

Abstract. Autonomy is a key feature in software agent design. For a system to function according to the expected behavior, however, the expectations must be expressed through a set of norms. The present work proposes a model for representing norms in multi-agent systems and for instantiating norms by using context-aware policies. The policies take into account both the general context and the reputation level of the target agent. The proposal aims at increasing the scalability and the interoperability of the normative systems implementing it, along with offering expressivity and richness in guiding agents towards the desired behavior.

1 Introduction

In guiding the actions of their members toward the desirable behavior, human societies grant some rights under the constraints of some boundaries. The boundaries may either be abided by the individual, or, the individual will bear the sanctions entailed by the infringement. The same strategy can be adopted in the electronic environment if provided the mechanisms to represent, communicate and enforce norms, and mechanisms to reason about norms and detect norm violation.

This paper proposes a normative model for open multi-agent systems, introducing a unified format for norm representation and a methodology that offers flexibility and scalability in what concerns norm enforcement.

Norms are needed for setting the expectation of what are the appropriate actions. As a social contract, they represent an agreement between the members of the community. The individual surrenders liberty in return for protection. Norms tell the individual what is appropriate to do in order not to lose utility and what to expect of the behavior of others. For this reason, computational normative systems are needed in industrial applications. They would allow business partners to have the correct

Andreea Urzică · Cristian Gratie
University Politehnica of Bucharest
e-mail: {andreea.urzica,cristian.gratie}@cs.pub.ro

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 287–296.
springerlink.com

expectations about each others actions and be free to choose their own way of fulfilling their obligations.

Normative systems help the members of a community have a common system of values to refer to and lead to an alignment in evaluation. The norms stand as a base for communication, if we think of languages as a collection of rules, or when we think of standard specifications.

The focus of this paper is set on the scalability resulting from the use of a well designed normative model. The model we propose allows using the same set of norms in two systems with different policies without losing the expressivity of the desired behavior. This feature allows the two systems to be interoperable as agents can migrate easily from one to another. The present paper stresses on the conceptual detachment between specifying the desirable behavior, by norms, and specifying the precise context-dependent parameters by policies. This approach reflects one of the main benefits of using norms within open and distributed multi-agent systems: interoperability between various normative systems of agent communities that share the same set of values.

The method proposed in this paper functions as the machinery that transforms a given set of generic norms into the actual clauses of a contract to be signed by two or more parties.

After discussing related work in Section 2 the paper describes the format for representing contracts and norms in Section 3 and policies in Section 4. Section 5 explains how policies are used to instantiate norms and produce contract pro-posals. Section 6 concludes the paper and indicates directions for future work.

2 Related Work

Nowadays, normative systems are the center of many research efforts (e.g. [2], [7], [3]). They aim at addressing the following open issues: how norms are learnt and how norms are propagated within a population, how to ensure norm compliance, who creates the norms, who assigns the norm, what phenomena can be captured by norm representation etc. The socialization aspect of the normative systems concerns the perception of the compliance of agent behavior with the norms, the evaluation of norm violations and applying sanctions. Norms are enforced by using incentives and sanctions, usually by endowing a valueless artifact with value (e.g. the monetary system, reputation). The present work provides ways of reducing the complexity in representing and communicating norms between different normative systems and offers the means for using reputation as a motivational factor in ensuring norm compliance.

As mentioned in [1], autonomy is a crucial notion in multi-agent systems. The authors of [1] add that there is still another aspect of the autonomy that should be considered when discussing the autonomous character of an agent: the context. They argue that an agent can be autonomous in a situation and non-autonomous in the other. For this reason the architecture we propose explicitly identifies, by the

use of policies, what are the areas that ensure the autonomous character of the agent while it activates in a norm-regulated community.

Many works, e.g. [3], consider the communication of facts together with reputation information of great relevance for the mechanisms that use reputation within normative contexts. This way, each agent has all the data needed to reason about norm violation or fulfillment, sanctions assignment and individual profiles. The reputation may be altered by norm violation and it may depend on the nature of the norm: breaking an organizational norm may be much more costly than breaking an individual norm (the rule norms and social norms as defined in [4]). Building on this differentiation, the system described in our paper allows agents to customize the reputation sanctions associated to public norm violations according to their own set of policies.

The proposed formalization of norms is inspired from that found in the L.I.A.R model [7], but, differently from L.I.A.R, it specifies the sanction inside the norm. The sanctions are specified using formal parameters that will be later instantiated by policies. In L.I.A.R, sanctions are assigned by the evaluators when generating a new social policy based on an observed interaction. In our model different evaluators are also able to assign different sanctions since each evaluator may have a different policy set. In addition, the same set of norms is acknowledged by all the participants. This aspect ensures that all agents refer to a common set of values.

A significant contribution to this research area has been brought by the Contract Project [6], [5] which aims at applying electronic contracting and contract-based monitoring techniques in real-world applications. The benefits of using contracts in modern business relationships are extensively reported in [6]. [6] provides four real business cases, arguing how electronic contracting is the most suitable way of regulating agent behavior for industrial applications. A simple reason is that 'it mirrors existing (non-electronic) practice, aiding adoption.' [6].

A valuable aspect that is missing from the framework presented in [6] is the distinction between normative roles and domain-specific roles within a contract. The model we propose benefits from using two different layers, the normative layer and the domain-specific layer, so that the normative roles are not associated directly to the individual, but to a class of individuals within the multi-agent sys-tem, namely, the domain-specific role.

The importance of working with both a contract template and a contract pro-posal is defended also in [6], but they explain nothing about how contract tem-plates are instantiated to contract proposals or on which bases the choices of con-tract management and update are made. The model proposed in the present paper focuses on the procedure (including formal representation and algorithm) of producing multiple contract proposals based on one contract template and a request from the business partner. This approach also covers the negotiation phase, since the resulting contract proposals may include all the possible combinations of values for the negotiable terms (e.g.: $proposal_1(price = 100, QoS = good), proposal_2(price = 50, QoS = medium)$). In addition the proposed model takes into account the reputation of the business partner, allowing the provider to adapt his offers based on his own set of policies, so that they reflect his trust level associated with the given reputation value.

3 Contracts and Norms

Within business relationships it is essential to explicitly describe each partys responsibilities with respect to a given interaction. Both parties need to make a commitment towards a set of indicators they have previously agreed upon. Without these normative means there will be little or no trust involved, the performance of each participant would not be properly evaluated at the end of the interaction and the output would be less than optimum.

The proposed model considers the need for a set of generic norms associated to each service, acknowledged by all the members of the system. For each service, all the *norms* describing the boundaries of the interaction are grouped into a set, called hereafter *contract*.

There are two types of contracts in the proposed model: *contract templates* and *contract proposals*. Contract templates describe the set of norms associated with a service and express the appropriate behavior for each actor of the interaction. The norms within contract templates use only variables, called hereafter formal parameters, which are not bound to literals. Contract proposals result from binding the formal parameters within the norms to literals decided by selecting and applying context-aware policies. The interaction between the service provider and service consumer will take place only if they agree on one contract proposal and commit themselves to comply with all the actual norms within that contract.

Definition 1. A contract template is defined as follows:

$$c_{template} < \mathscr{W}, \mathscr{D}, \delta, \mathscr{R}, \mathscr{N}, >$$

where,

- $\mathscr{W}$ is a (finite)set of formal parameters;
- $\mathscr{D}$ is a (finite) set of domains (sets of values);
- $\delta : \mathscr{W} \rightarrow \mathscr{D}$ is a function that maps each variable to its domain (multiple variables may have the same domain);
- $\mathscr{R}$ is the (finite) set of roles specific to the interaction type (service);
- $\mathscr{N} = \{\nu_1, \nu_2, ..., \nu_p\}$ is a (finite) set of norms, where each **norm** ν_i is itself defined as a tuple:

  $$\nu_i < \omega_i, \tau_i, \varepsilon_i, \alpha_i, \gamma_i, >$$

  such that,

  - $\omega_i \in \Psi$ is a deontic operator applied to the content of the norm;
  - $\Psi = \{Interdiction, Permission, Obligation\}$;
  - $\tau_i \in \mathscr{R}$ is the target role,the entity the norm is addreesed to;
  - $\varepsilon_i \in \mathscr{R}$ is the evaluator role,entitled to evaluate whether the norm was respected or not (and possibly to apply sanctions to the target);

- $\alpha_i = A_i(V_{i1}, V_{i2}, ..., V_{ik_i})$, α_i is a k_i-ary predicate that describes the pertinence (applicability)condition of the norm;
- $\gamma_i = C_i(V'_{i1}, V'_{i2}, ..., V'_{ik_i})$, γ_i is a k'_i-ary predicate that describes the content of the norm;
- $V'_{ij} \in \mathscr{W}$ for all $j = \overline{1, k'_i}$;

The *pertinence condition* within a norm is a predicate that can decide whether the norm is applicable or not in the current context of the system. Unlike an acti-vation condition, pertinence controls both the activation and the deactivation of the norm. The norm is applicable only as long as the pertinence condition holds. Should the norm be applicable, its outcome is represented by its content coupled with the deontic operator.

The *norm content* is a predicate and it tests the actions of the agent that plays the normative role called *target*. The agent with the *evaluator* role is the one that is entitled to actually check if the content of the norm holds while the pertinence condition holds. The evaluator may also apply sanctions to the target. Thus, in the present model, the evaluators represent both the judiciary and the executive power. A discussion about the legislator, namely the entity endowed with the power of issuing the generic norms is outside the scope of this paper.

The norms, in the proposed model, are the means of assigning to each actor of the interaction the responsibilities associated to its position. Any interaction-specific role may be attributed the *target* normative role. For example, in the case of a commercial interaction, either the seller or the buyer may be considered target within a norm. For the more general case, where all types of interactions are modeled as services, either the provider or the consumer of the service may be considered target within a norm. The proposed model clearly separates the two layers: the normative layer (here the roles are target and evaluator) from the multi-agent layer (where the roles may be seller, buyer, student, teacher etc, depending on the interaction type).

The BNF definition of a norm in the proposed model is presented below.

```
  <Norm> ::= <deonticOp> <Target> <Evaluator>
<PertinenceCondition> <Content>
  <deonticOp> ::= 'I' | 'O' | 'P'
  <Target> ::= <RoleName>
  <Evaluator> ::= <RoleName>
  <PertinenceCondition> ::= <IntegrityConstraint> |
<Condition> | <Authorization> | <Timer>
  <Content> ::= <Fact>
```

The content of a norm, as well as the pertinence condition, is formed only by using formal parameters (variables). This provides a high degree of generality and thus scalability. The same set of norms may be adopted by a broader community of agent systems without losing expressivity in reflecting the desirable behavior.

As an illustration, we can refer to the International Road Traffic Regulations. The desired behavior is that drivers may cross the dashed line between lanes but they may not cross the continuous line mark between lanes. The norm should express the sets of entities playing the role of target and evaluators, the condition under which the norm content becomes applicable and content of the norm. The norm may be formulated as 'it is forbidden to cross the continuous line' or 'it is permitted to cross the dashed line' or 'the driver who has crossed the continuous line is obliged to pay a fine', etc. However, a norm should not contain explicitly the amount of money representing the fine. The numerical value of the fine may be established differently by each country and still, the same norm will be followed by all the countries in the community. This approach allows agents to easily adapt when traveling to other countries because there is not a new behavior they have to adopt, only different values for the already acknowledged parameters.

The actual values (e.g. numerical values) associated to the formal parameters within a norm are to be set by using policies. The set of all formal parameters within the set of norms may be seen as the set of business rules terms defining the interaction. For the case of a commercial interaction, e.g. booking a hotel room, the set of business rule terms include the price per night for a room, the hour for check in, the hour for check out, etc. The norms would specify that 'the client is obliged to pay the hotel the price per night multiplied by the number of nights' or that 'the client is permitted to check in after the check-in hour', etc, but it is each hotels *policy* that establishes the actual values for these business terms. The next section provides the formal description and explains the use of the *policy* concept.

4 Policies

The process of instantiating contract templates into actual contract proposals makes use of policies. By using policies, this process can take into account the context of each particular interaction and the reputation of the participants. Policies are dictated by the business rules of each entity and are used in order to interpret the given context and assing actual values to all the variables specified in the contract.

Definition 2. A policy with respect to a contract template $< \mathscr{W}, \mathscr{D}, \delta, \mathscr{R}, \mathscr{N} >$ is a set of mappings that, given a reputation value and known variables, returns possible values for all variables:

$$\pi = \{\pi_k : \mathbf{R} \times \Pi_{V \in K} \delta(V) \rightarrow \mathscr{P}(\Pi_{V \in \mathscr{W}} \delta(V)) | K \in \mathscr{P}(\mathscr{W})\}$$

where, for a given set X, $\mathscr{P}(X)$ stands for all subsets of X.

The BNF definition of a policy in the proposed model is presented below.

```
  <Policy> ::= <Match> <Action>
  <Match> ::= <Condtion>+
  <Condition> ::= <FormalParam> <Relation>
<Literal>
  <FormalParam> :: = <BusinessTerm> | <Type>
  <Relation> ::= '<'| '='| '>'| "<=" | ">="
  <Action> ::= <ActualParamList>
  <ActualParamList> ::= <Literal>+
  <Request> ::= <Preference>+
  <Preference> ::= <Literal> | "N/A"
  <Context> ::= <Request> <ClientData> <Reputation>
<Situation>
  <Situation> ::= <Literal>+
  //describes the current state of the service
provider and the general context
  <ClientData> ::= <Literal>+
  // ClientData contains complementary information
about the agent (e.g. age, nationality, credit card
expiry date, the type of identification document,
the date of obtaining the drivers license, etc.)
  <Reputation> ::= <RepValue>
  //the reputation of the agent requesting the
service
  <RepValue> ::= <Literal>
```

The proposed model considers all interaction types to be modeled as services. The list of literals within a *request* must be formed in conformity with service specifications. The order and the semantic of the literals in a request are defined in service description.

The Reputation field is considered separately, as it is a constant factor to be taken into account by any policy with regard to the target of the norm. In addition we considered important to indicate *Reputation* separately in order to highlight the link between the normative system and a reputation model that could be used for norm enforcement.

Each agent has a set of policies in order to capture all the relevant cases and combinations of context factors and reputation values. A provider with no flexibility would have only one set of literals associated to the actual parameters list, regardless of the context or the reputation of the client. Since some agents may want to offer better prices for the same service to clients with good reputation in order to apply incentives for norm compliance, the present model provides the means for this. The process is depicted in Fig. 1.

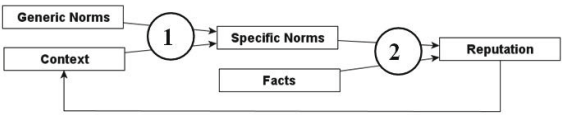


Fig. 1 The interdependency between reputation and norms

The first step depicted in Fig. 1 represents the action of applying the policies, filtered by context, to the set of generic norms in order to obtain instantiated contracts. Step 2 shown in Fig. 1 represents the action of testing the facts against a certain set of instantiated norms (agreed upon) and updating the reputation associated to the target of those norms.

5 Instantiating Norms by Using Policies

This section explains the procedure of issuing contract proposals by choosing the corresponding set of policies and applying them to norms. Policies are applied to the generic norms in order to generate specific, context-aware norms, as depicted in Fig. 2.

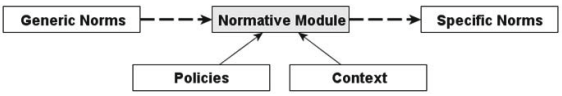


Fig. 2 The normative system

Considering the interaction between a service provider and a consumer, both having had acknowledged the same contract template, the steps for instantiating the contract are the following.

The consumer sends a request to the service provider, containing the set of requirements for the service. The request consists in specific values for some of the formal parameters of the contract, representing the list of preferences of the consumer regarding the given service.

The service provider identifies the policies that are applicable for the context described by the clients request. Then the provider retrieves the reputation of the consumer and selects the policies that apply for the clients reputation level.

By applying the actual parameter list resulted from the selected policies to the contract template, one or more versions of instantiated contracts result. These versions will be called *contract proposals* hereafter.

The provider sends the contract proposals to the consumer. The consumer evaluates them and chooses one. The contract proposal chosen and signed by the consumer represents the instantiated contract that will further regulate the interaction. The chosen contract proposal contains the set of norms to which both parties have committed themselves to comply. The norm instantiating algorithm is presented below.

1. The *ActualParamList* is initialized to an array default, consisting of default parameters, corresponding to the default policy.
2. The literal array, representing an agents request, is appended with other parameters (i.e. literals) describing: agent data (e.g. age, nationality, etc.), agent reputation and a series of literals specific to the service provider in the given context (e.g. for the case of a car-rental company, how many cars are available that day, weather conditions, etc).
3. The resulted array is named *context*.
4. From the array named *context* and on the basis of service implementation specifications the difference between the two moments in time provided in the request is identified, and the resulting period is divided by the greatest time unit that allows for pricing the service (e.g. a day).
5. For each taxing unit (e.g. per day) resulted from the context, a replica of the context is created.
6. For each context replica, the policies in the list of policies are selected one by one, (following the order they were provided) and for each one it is checked if the literals validate the conditions or not.
7. When, by replacing all the variables from the matching part with the given literals, no false value results, the policy is selected in order to be applied.
8. All the literals from the *Action* part of the policy replace the formal parameters within the norms in the contract template.
9. In some cases additional processing is necessary in order to derive terms. In such situations, derivation rules are applied (e.g. the price for a longer period is calculated as the sum of the prices for each day).
10. Derivation rules are mathematical formulas for computing specific business parameters from given parameters.

6 Conclusions

The present paper proposes a unified format for norm description within multi-agent systems. The proposed method aims at reducing the complexity of communicating norms and reasoning about norm enforcement while offering flexibility and variety in issuing business contracts.

The normative system described here is conceived as being ready to be linked with a reputation mechanism, so that norm enforcement can be attained by influencing reputation information. Agents may use policies that encourage norm compliance and penalize violations in various degrees. The emergent norm can be observed in this case: the cessation of interactions between untrustworthy entities and honest entities because of the disadvantageous conditions requested for a new interaction.

Such a model combines in a realistic manner the norms with a sanctioning system (reputation) that has proved to be efficient in the business environment.

The proposed model is intended to serve as a basis for any further evolution, being designed in a modular manner and using a separate and clearly defined normative layer. Since any business interaction can be modeled as a service, we argue that the normative system described in this paper that can be employed on a rich set of application domains.

References

1. Carabelea, C., Boissier, O., Florea, A.: Autonomy in Multi-agent Systems: A Classification Attempt. In: Nickles, M., Rovatsos, M., Weiss, G. (eds.) AUTONOMY 2003. LNCS (LNAI), vol. 2969, pp. 103–113. Springer, Heidelberg (2004)
2. Conte, R., Paolucci, M.: Reputation in artificial societies: Social beliefs for social order. Kluwer Academic Publishers (2002) ISBN 978-1-4020-7186-7
3. da Silva, V.T., Hermoso, R., Centeno, R.: A Hybrid Reputation Model Based on the Use of Organizations. In: Hübner, J.F., Matson, E., Boissier, O., Dignum, V. (eds.) COIN@AAMAS 2008. LNCS, vol. 5428, pp. 111–125. Springer, Heidelberg (2009)
4. Hermoso, R., Centeno, R., da Silva, V.: Reputation-based Agreement for Agent Organisations. In: Proceedings of the Second Workshop on Agreement Technology at Conferencia de la Asociación Espanola para la Inteligencia Artificial (CAEPIA), pp. 82–93 (2010)
5. IST CONTRACT project (2008), http://www.ist-contract.org/
6. Jakob, M., Pěchouček, M., Miles, S., Luck, M.: Case studies for contract-based systems. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems: Industrial Track, pp. 55–62 (2008)
7. Vercouter, L., Muller, G.: L.I.A.R.: Achieving Social Control in Open and Decentralised Multi-Agent Systems. Applied Artificial Intelligence 24(8), 723–768 (2010)

A Multi-Agent System for Service Acquiring in Smart Environments

Irina Mocanu, Lorina Negreanu, and Adina Magda Florea

Abstract. This paper proposes a multi-agent system architecture for a service acquiring system. The proposed system is integrated in an ambient intelligent system, AmIHomCare, a smart house who supervises elderly people in their homes and also helps people during their daily activities. The service acquiring system is specified and validated using a formal specification method - Event B.

1 Introduction

The percentage of elderly in today's societies keeps on growing. As a consequence we are faced with the problem of supporting older adults in loss of cognitive autonomy who wish to continue living independently in their home as opposed to being forced to live in a hospital. Smart environments have been developed in order to provide support to the elderly people or people with risk factors who wish to continue living independently in their homes, as opposed to live in an institutional care. The term of Ambient Intelligence (AmI) was introduced to describe a ubiquitous electronic environment that would pro-actively, but sensibly and non-intrusively support people in their daily lives [9].

Our goal is to specify and validate using a formal specification method a multi-agent system for acquiring services that will be used in an intelligent house (called AmIHomCare). The AmIHomCare system performs home monitoring and assisting of the elderly people or patients with risk factors. Thus we use the formal method Event-B in order to prove the correctness of the whole process of service acquiring.

The rest of the paper is organized as follows. Section 2 describes the proposed service acquiring system integrated in the AmIHomCare system. Section 3 presents the formal specification and verification of the proposed system. Conclusions and future works are listed in Section 4.

Irina Mocanu · Lorina Negreanu · Adina Magda Florea
University "Politehnica" of Bucharest, Computer Science Department,
Splaiul Independentei 313, 060042 Bucharest, Romania
e-mail: {irina.mocanu,lorina.negreanu,adina.florea}@cs.pub.ro

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 297–306.
springerlink.com

2 The Service Acquiring System

In [8] the AmIHomCare system was presented. The system is an intelligent house for home medical assistance of elderly or disabled people. The main objectives of this system are: (i) performing home monitoring and assistance for elderly people and (ii) detecting the medical emergencies and providing medical assistance. One component of the AmIHomCare system is The Multi-Agent System for Information Ac-cess component (MASIA) who achieves pervasive information access and retrieval based on captured images and patient specific context. The MASIA component consists of the patient supervising system and the service acquiring system.

The supervising system analysis the images received from a supervision cam-era. Each image together with data provided by different sensors (movement sensor and sonar) are analyzed in order to perform human activity recognition. First the context of the supervised person (its location in the room and its body posture) is obtained. Each obtained context represents a sub-activity. A sequence of sub-activities forms a recognized activity. The data resulted from this processing is stored as monitoring information. The monitoring information together with the daily activity program for the supervised person will be analyzed with the purpose of detecting any emergency situation, or other situation which requires assistance. In either case, a message is sent to the call center or to the home assistance center.

Sometimes the supervised person wants to perform different activities outside of his home. Thus we consider a relaxing center that is able to offer different types of activities, for example: swimming, physiotherapy or simply socializing. The user will request a type of service for a specific duration, for example half an hour of physiotherapy. We assume that a request contains only one reference to a service of certain duration. Thus the system will verify if the request can be satisfied. A request can be either satisfied or pending: a request is satisfied if it is scheduled; it remains pending, if it cannot be scheduled.

Each type of service has an associated maximum (aggregated) duration. The queries are satisfied if the requested service has enough free time to satisfy that request. The type of the desired service will be specified as an image query. The service acquiring system will analyze the image query, detect the relevant object from the query and make a request for that service with the specified duration. The properties of the possible services together with relevant images are saved in the object ontology. The requested duration for a service is computed by the service acquiring system using the saved monitoring information.

We use the architectural description of the supervising system described in [7] and we add a new layer to integrate the service acquiring system in the MASIA component. Thus the architecture of the MASIA component consists of five layers. Each layer is composed of a set of agents. The architecture of the MASIA component is described in Fig. 1.

The *information gathering* layer consists of a set of sensor agents: an agent to provide information (Video Camera Agent), an agent to interrogate the ambient factors (Temperature Agent) and an agent to verify the health status of the supervised person (Pulse Agent). The person's context is composed of its localization

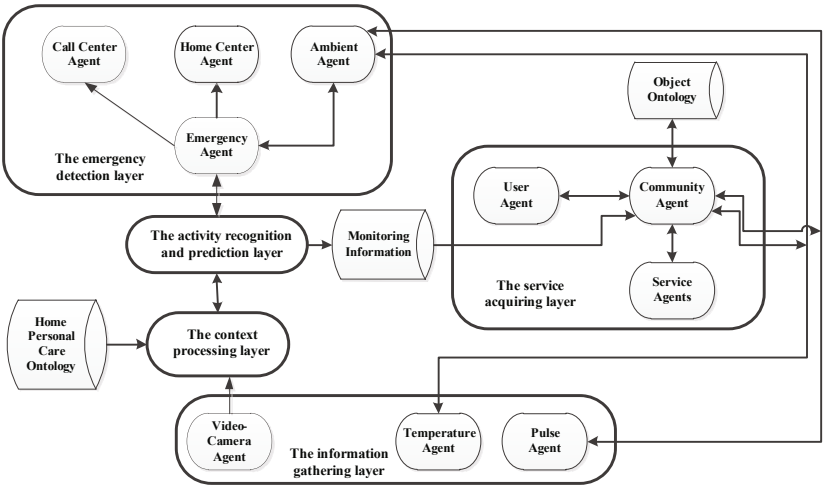


Fig. 1 The multi-agent system for the MASIA component

in the room together with the surrounding furniture objects from the room and its body posture. The agents from the *context processing layer* have access to the room model (represented by the Home Personal Care Ontology). The *activity recognition and prediction layer* achieves daily activity recognition of the supervised person. Also this layer is responsible with prediction of the next activity, [6]. The *emergency detection layer* performs emergency detection. Its goal is to detect different situations which correspond to a high or a low level emergency and send messages to the home assistance center or to the call center. The *service acquiring layer* connects user with a requested service (if the service is available). It contains a set of Service Agents - each agent is responsible with a specific service, a Community Agent which knows all the services (available and unavailable).

The service acquiring system contains: the User Agent, the Community Agent and the Service Agents. All the requested service process is shown in Fig. 2. The Service Agents are agents associated to all known services: available or unavail-able. If a service is available it will have associated a maximum available time. All the Service Agents are recorded to the Community Agent. The user makes a query using an image. The User Agent will analyze the image, detect the type of the requested service (based on the object properties from the Object Ontology) together with its recommended duration. The recommended duration is obtained analyzing the monitoring information extracted from the daily activities. Thus a request service will be sent to the Community Agent (*new_request* with id - user identifier, type - type of the requested service and duration - requested duration). The Community Agent will send the request both to the Ambient Agent and also to the Pulse Agent in order to verify if the supervised person can perform that activity.

For example it is a hot summer day and the supervised person wants to meet with somebody else. The Ambient Agent will detect the temperature and will send a *cancel_request* to the Community Agent which will cancel the requested service. Also the Pulse Agent will verify the health status of the supervised person. For example he requested a swimming process but he has a high pulse. In this case the Pulse Agent will send a *cancel_request* to the Community Agent which will cancel the requested service.

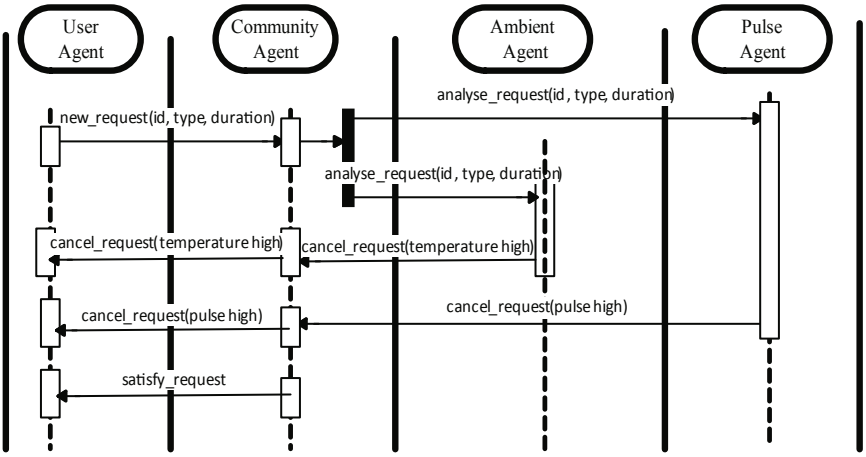


Fig. 2 The sequence diagram for the service acquiring process

In this case the Pulse Agent will send a cancel request to the Community Agent. After that the Community Agent will interrogate all the service agents (of the requested type). If a service is available the user will obtain the acceptance for the service (*satisfy_request*). Some services may be requested more frequently than others. Thus the Community Agent may modify the type for a Service Agent (*add_to_available*).

3 Formal Verification for the Service Acquiring System

Event-B is a formal method for system-level modeling and analysis. Key features of Event-B are the use of set theory as a modeling notation, the use of refinement to represent systems at different abstraction levels and the use of mathematical proof to verify consistency between refinement levels [1]. An Event-B based formal model is centered around the concepts of variables, events and invariants; variables describe the state of the system and can be modified by events; invariants state some

properties that must always be satisfied by the variables and maintained by the activation of the events. Each basic object is a set; relations and functions should be considered primarily as sets [3], [5].

From the modeling point of view a request is a member of a set, namely the set of *requests*. *To satisfy a request* is an event which modifies the status of the request from pending to *satisfied*. In our model the event is called *satisfy_request* and is triggered for each pending request. The satisfied requests are represented by the set called *satisfied_requests* (a subset of the set of existing requests); the pending requests are defined by the complement set of *satisfied_requests* with regard to the set of existing requests.

We introduce the abstract set of all possible requests ALL_REQUESTS (existing and future) for which the existing requests is a subset: $requests \subseteq$ ALL_REQUESTS. The request is the main modeling element. We can access the service and the duration associated with a request, through the following functions: $reference \in requests \rightarrow$ SERVICES where *reference* assigns a service to each request; $duration \in requests \rightarrow N^*$ where *duration* assigns a duration to each request (with the assumption that if a service is requested, the duration is at least 1).

The state of the request is changed into *satisfied*, if the request is scheduled, i.e. the duration of the requested service is either less than or equal to the time availability of the service. We will call the time availability of a service *available*: $available \in SERVICES \rightarrow N$.

In our modeling we start by considering that all requested references exist in the set of available services and that this set and the set of requests are given in an up-to-date state. We derive by refinement [4] the situation in which we have to take into account new requests, cancelation of requests and increase of services time availability. The first Event-B model is specified bellow.

CONTEXT
Services_c0
SETS
ALL_REQUESTS
SERVICES
END
MACHINE
Services_0
SEES
Services_c0
VARIABLES
requests
available
satisfied_requests
duration
reference
INVARIANTS
inv1 : requests $\subseteq$ ALL_REQUESTS
inv2 : available $\in$ SERVICES $\rightarrow$ N

inv3 : satisfied_requests $\subseteq$ requests
inv4: duration $\in$ requests $\rightarrow$ N1
inv5: reference $\in$ requests $\rightarrow$ SERVICES
EVENTS
INITIALISATION:
THEN
act1: requests := $\emptyset$
act2: available := SERVICES $\times$ {0}
act3: satisfied_request := $\emptyset$
act4: duration := $\emptyset$
act5: reference := $\emptyset$
END

satisfy_request:
ANY
r
WHERE
grd1: r $\in$ requests $\setminus$ satisfied_requests
grd2: duration(r) $\leq$ available(reference(r))
THEN
act1: satisfied_requests := satisfied_requests $\cup$ {r}
act2: available(reference(r)) := available(reference(r))- duration(r)
END

cancel_request:
THEN
act1 : requests, satisfied_requests, duration, reference: | requests' $\subseteq$ ALL_REQUESTS
$\wedge$ satisfied_requests' $\subseteq$ requests' $\wedge$ duration' $\in$ requests' $\rightarrow$ N1
$\wedge$ reference' $\in$ requests' $\rightarrow$ SERVICES
END

new_request:
THEN
act1: requests, satisfied_requests, duration, reference:| requests' $\subseteq$ ALL_REQUESTS
$\wedge$ status' $\in$ satisfied_requests' $\subseteq$ requests' $\wedge$ duration' $\in$ requests' $\rightarrow$ N1
$\wedge$ reference' $\in$ requests' $\rightarrow$ SERVICES
END

add_to_available:
THEN
act1: available : | available' $\in$ SERVICES $\rightarrow$ N
END
END

An Event-B model encapsulates variables defining the state of the system; the state should conform to the invariant and each event can be triggered when the current state satisfy the invariant. The static elements of the model are specified through the context component; the dynamic behavior of the model is specified through

the machine component. A context may have the components SETS, CONSTANTS and AXIOMS; the clause SETS contains definitions of user defined types; the clause CONSTANTS allows the declaration of constants, and the clause AXIOMS contains a list of predicates that define rules for the given elements of the context; each axiom has a label attached to it. A machine describes the dynamic behavior of a model by means of variables whose values are changed by events. A central aspect of modeling a machine is to prove that the machine never reaches an invalid state, i.e. the variables always have values that satisfy the invariants. A machine may contain the following components: REFINES - a machine has the option of refining another one; SEES - the context's sets, constants and axioms can be used in a machine; the axioms can be used in every proof in the machine as hypotheses; VARIABLES - the variables' values are determined by an initialization event and can be changed by events; this constitutes the state of the machine; the type of each variable must be declared in the invariant section; INVARIANTS - are predicates that should be true for every reachable state; each invariant has a label; EVENTS - an event can assign new values to variables; the guards of an event specify the conditions under which it can be executed; the initialization of the machine is a special case of an event [2].

Conditions of verification called proof obligations are generated from the text of the model using SETS, CONSTANTS and AXIOMS clauses for defining the mathematical theory and the INVARIANT, and INITIALIZATION clauses to generate proof obligations for the preservation of the invariant and proof obligations stating the correctness of safety properties with respect to the invariant [2].

In order to specify the events we consider that all requested references are references in *available* and the *satisfy_request* event is triggered when the time availability of the requested service is greater than the duration requested. Let r be a pending request ($r \in requests \wedge status(r) = pending$). If the time availability of the service whose reference is $reference(r)$ is greater than the duration of the request $duration(r) \leq available(reference(r))$, then the request is satisfied: $status(r) = satisfied$ and the time availability of the service is decreased: $available(reference(r)) := available(reference(r)) - duration(r)$.

The event *modify_request* allows the modification of the duration of a request, if that request is still pending. Let r be a pending request ($r \in requests \wedge status(r) = pending$), and d the new duration, then the duration of the request is updated: $duration(r) = d$.

The events *cancel_request*, *new_request*, and *add_to_available* are modeling the state changes for the variables attached to the time availability of services and requests. We do not know how these variables are modified therefore we express that a modification is possible, using the non-deterministic assignment :— with a before-after-predicate ($x_1, \ldots, x_n :\mid Q(c, w, u, x_1^{'}, \ldots x_n^{'})$ assigns any value to the variables $x_1, \ldots, x_n$ such that the the before-after-predicate Q is fulfilled).

We refine the first model by taking into account new requests, cancelations of requests and increase of service's time availability. The last events of the abstract model should be refined to handle the modifications of the variables *requests*, *status*, *duration*, *reference* and *available*. The *new_request* event modifies *requests*, *status*, *duration*, and *reference*; it adds a new request called r which

is not existing in the current set of requests; its status is *pending*; *duration* and *reference* are updated according to the requested duration d and reference s. The *cancel_request* event modifies *requests*, *status*, *duration*, *reference*; it removes a request called r which is pending in the current set of requests; *status*, *duration* and *reference* are updated by using the domain-subtraction operator ($\ll|$): $status := \{r\} \ll| status, duration := \{r\} \ll| duration, \{r\} \ll| reference$. The *add_to_available* event adds a given duration d for a given service s. The model follows:

MACHINE
Services_1
REFINES
Services_0
SEES
Services_c0
VARIABLES
requests
available
satisfied_requests
duration
reference
EVENTS
INITIALISATION:
THEN
act1: requests := $\emptyset$
act2: available := SERVICES $\times$ {0}
act3: satisfied_request := $\emptyset$
act4: duration := $\emptyset$
act5: reference := $\emptyset$
END

satisfy_request:
REFINES
satisfy_request
ANY
r
WHERE
grd1: r $\in$ requests $\setminus$ satisfied_requests
grd2: duration(r) $\leq$ available(reference(r))
THEN
act1: satisfied_requests := satisfied_requests $\cup$ {r}
act2: available(reference(r)) := available(reference(r))- duration(r)
END

cancel_request:
REFINES
cancel_request
ANY
r

```
    WHERE
      grd1: r ∈ requests \ satisfied_requests

    THEN
      act1: requests = requests \ {r}
      act2: duration = {r} ⩤ duration
      act3: reference = {r} ⩤ reference
  END

 new_request:
    REFINES
      new_request
    ANY
      r
      d
      s
    WHERE
      grd1: r ∈ ALL_REQUESTS \ requests
      grd2: d ∈ N1
      grd3: s ∈ SERVICES
    THEN
      act1: requests := requests ∪ {r}
      act2: duration(r) := d
      act3: reference(r) := s
    END

  add_to_available:
    REFINES
      add_to_available
    ANY
      d
      s
    WHERE
      grd1: d ∈ N1
      grd2: s ∈ SERVICES
    THEN
      act1: available(s) := available(s) + d
    END
  END
```

The models have been specified and validated using Rodin [1], an Eclipse-based IDE for Event-B that provides support for refinement and mathematical proofs. A model is validated by discharging proof obligations. The statement of development is described in the table 1 with the required proof obligations.

Table 1 The statement of the development

Element_name	Total	Auto	Manual	Reviewed	Undischarged
Services_0	18	15	3	0	0
Services_1	5	5	0	0	0

4 Conclusions and Future Works

In this paper, we have presented a specification and verification technique of a multi agent system. The informal specification of the system is translated into the Event-B notation to verify required properties. The model refinement that Event-B emphasizes simplifies proofs by providing a progressive and detailed view of the system. We can further refine our model, by taking into account the flow of re-quests. The presented model captures the notion of flow set, meaning that the ordering of arrival is not expressed. In fact it is possible that a request remains al-ways pending and is never satisfied, because there are always other requests which are processed. As future work we add a criterion to each request in order to sort them. Also we will verify the whole MASIA component.

Acknowledgements. The work has been co-founded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557 and FP7 project ERRIC: Empowering Romanian Research on Intelligent Information Technologies, No. 264207.

References

1. Rodin platform, http://wiki.event-b.org/index.php/Rodin_Platform
2. Rodin user's handbook v.2.5., http://handbook.event-b.org/current/html/index.html
3. Abrial, J.-R.: The B book (1996)
4. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering (2010)
5. Abrial, J.-R., Cansell, D., Méry, D.: Refinement and Reachability in Event_B. In: Treharne, H., King, S., C. Henson, M., Schneider, S. (eds.) ZB 2005. LNCS, vol. 3455, pp. 222–241. Springer, Heidelberg (2005)
6. Das, S.K., Cook, D.J.: Designing Smart Environments: A Paradigm Based on Learning and Prediction. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 80–90. Springer, Heidelberg (2005)
7. Mocanu, I., Florea, A.M.: A multi-agent supervising system for smart environments. In: Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics – WIMS 2012. ACM (2012)
8. Mocanu, S., Mocanu, I., Anton, S., Munteanu, C.: AmIHomCare: A complex ambient intelligent system for home medical assistance. In: Proceedings of the 10th International Conference on Applied Computer and Applied Computational Science, pp. 181–186 (2011)
9. Ramos, C., Augusto, J., Shapiro, D.: Ambient intelligence - the next step for artificial intelligence. IEEE Intelligent Systems 23(2), 15–18 (2008)

Newtonian Emotion System

Valentin Lungu

Abstract. The main goal of our work is to provide virtual characters with the ability of emotion expression. Our approach is based on the idea that emotions serve a fundamental function in human behavior, and that in order to provide artificial agents with rich believable affective behavior, we need to develop an artificial system that serves the same purpose. We believe that emotions act as a subsystem that enhances human behavior, by stepping up brain activity in arousing circumstances, directing attention and behavior, establishing importance of events and act as motivation. This paper summarizes the psychological theories that our computational emotion model is based on, defines the key concepts of the Newtonian emotion model that we developed and describes how they work within an agent architecture. The Newtonian emotion model is a light-weight and scalable emotion representation and evaluation model to be used by virtual agents. We also designed a plug-and-play emotion subsystem to be integrated into any agent architecture.

1 Introduction

Emotions are a part of the human evolutionary legacy [2], serving adaptive ends, acting as a heuristic in time-critical decision-processes (such as fight-or-flight situations); however, emotions have also become an important part of the multi-modal process that is human communication, to the point that its absence is noted and bothersome.

Affective computing is the study and development of systems that can recognize, interpret, process and simulate human affects, in order to add this extra dimension into our interaction with machines. Our goal is to create believable intelligent artificial characters capable of displaying affective behavior. To achieve this, we attempt to provide artificial characters with an emotional layer similar to that of humans,

Valentin Lungu
University Politehnica of Bucharest, Splaiul Independentei 313, 060042
e-mail: valentin.lungu.aimas@gmail.com

G. Fortino et al. (Eds.): Intelligent Distributed Computing VI, SCI 446, pp. 307–315.
springerlink.com

serving adaptive ends. The emotion subsystem will influence agent behavior by establishing the importance of events and by influencing knowledge processing, as well as provide the agent with an emotional state that it will be able to express and that will further influence its behavior. We first describe our Newtonian emotion system, that explains the way we represent emotions and how external and internal forces act upon them, based on the work of psychologist R.E. Plutchik [5] and Newton's laws of motion. Based on the psychological theory of dr. R. Lazarus [4], we devised an emotional subsystem that interacts with and influences an agent architecture. The system also takes into account the effects that emotions on human perception and cognitive processes.

As an application domain, we chose virtual characters in computer role playing games (CRPGs) as they are a type of game meant to simulate all manner of human interactions (be they gentle or violent in nature). This type of game provides us with complex dynamic environments in which modern agents are expected to operate. It also focuses on individual characters and the way they respond to relevent changes in their environment (behavior). Last, but not least, the limited action set available to characters and current generation game engines significantly reduce the overhead of proof-of-concept apps.

Applications of the technology are not limited simply to CRPGs, as several types of endeavours find it useful, such as behavior simulation, interactive storytelling, and, indeed, any application involving human-computer interaction (for example, online social sites could recommend events and friends based on mood).

2 Psychological Basis

In order to endow virtual characters with an emotion system that provides agents with believable emotional reactions and influences their cognitive processes (perception, decision making) in the same way that emotions influence behavior in humans, we need a grasp of human cognitive emotion theory. In this chapter we summarize the psychological theories upon which we built our emotion simulation model.

Plutchik

One of the most influential classification approaches for general emotional responses was developed by Robert Plutchik in the 1980s. Plutchik developed a theory showing eight primary human emotions: joy, trust, fear, surprise, sadness, disgust, anger, and, interest, and argued that all human emotions can be derived from these [5]. Plutchik also stated that emotions serve an adaptive role in helping organisms deal with key survival issues posed by the environment, and that the eight basic emotions evolved in order to increase the reproductive fitness of the individual, each triggering a behavior or type of behavior with a high survival value (such as fear and anger triggering fight or flight behaviors). These eight basic emotions were grouped into four pairs of polar opposites with varying degrees of intensity (displayed as the

wheel of emotions in Fig. 1), making it easy to represent emotional state in a four dimensional vectorial space.

Lazarus

In the 1960s, Singer-Schachter conducted a study showing that subjects can have different emotional reactions despite being in the same physiological state. During the study, experiments were performed where the subjects were placed in the same physiological state with an injection of adrenaline and observed to express either anger or amusement depending on whether a planted peer exhibited the same emotion. The study validated their hypothesis (known as the Singer-Schachter theory) that cognitive processes, not just physiological reactions, play an important role in determining emotions [6].

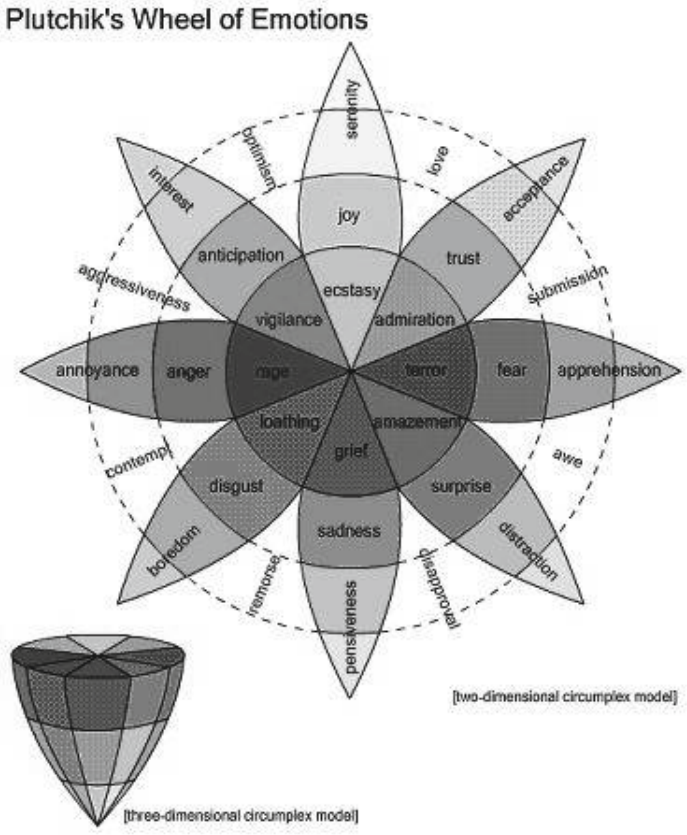


Fig. 1 Plutchik's wheel of emotions

A pioneer in the study of emotions and their relationship to cognition, Richard Lazarus, developed an influential theory in the field that explains the steps of emotion synthesis and how it affects cognition. As Singer-Schachter, Lazarus argued that cognitive activity (judgements, evaluations) is necessary for emotions to occur. He stressed that the quality (i.e. the combination of basic emotions according to Plutchik's theory) and intensity of emotions are controlled through cognitive processes, thus, the core of Lazarus' theory is the cognitive process of appraisal: before emotion occurs, people make an automatic, often unconscious assessment of the situation and its significance, and described the process as a three stage disturbance [4] (Fig. 2):

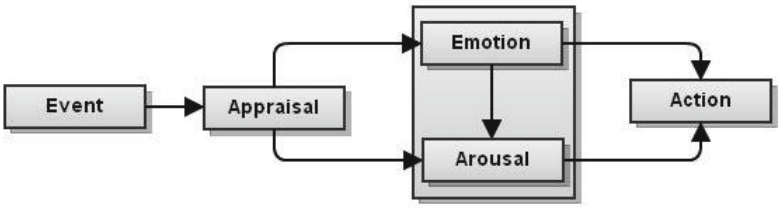


Fig. 2 Lazarus' emotion synthesis model

1. **appraisal** - the subject evaluates the event cognitively (consciously or not, symbolic or not)
2. a. **emotion** - cognitive appraisal triggers emotional changes (e.g. fear)
   b. **arousal** - the cognitive reaction coupled with the emotional response trigger physiological changes (e.g. increased heart rate and adrenal response)
3. **action** - the subject feels the emotion (which provides additional information) and chooses how to react

Perception

Emotions emerged in the process of evolution as the means by which living creatures determine the biological significance of the various states of the organism as well as of external influences; as a critical tool for adaptation and survival, emotions influence how we process and store information [7]. Through the attentional blink paradigm, it has been shown that events and objects that hold a high level of emotional arousal for the subject are more likely to be processed in conditions of limited attention [3], suggesting prioritized processing of such stimuli. What this means is that subjects will focus on the arousing details of the stimuli which will be processed, while peripheral details may not be (if attention is limited this leads to attention narrowing, where the subject's range of stimuli to which it is sensitive is decreased) [1].

3 Newtonian Emotion Space

In this chapter we will explain the basics of our Newtonian emotion representation system. We subscribe to Plutchik's theory (Sec. 2) in that only eight primary emotions are needed in order to derive the whole human emotional palette and that these eight primary emotions are grouped into four pairs of polar opposites, therefore, we chose to represent an emotional state as a vector in a four-dimensional space with the following four axes: joy-sadness, trust-disgust, anger-fear, and, anticipation-surprise.

Concepts

We will also introduce the following concepts in order to allow emotional states to interact with each other and external factors:

Definition 1. Position specifies the intersection of an emotional state with each of the four axes

Definition 2. Distance ($||\mathbf{p_2} - \mathbf{p_1}||$) measures the distance between two emotional *positions*

Definition 3. Velocity ($\mathbf{v} = \frac{\mathbf{p_2} - \mathbf{p_1}}{t}$) represents the magnitude and direction of an emotion's change of *position* within the emotion space over a unit of time

Definition 4. Acceleration ($\mathbf{a} = \frac{\mathbf{v_2} - \mathbf{v_1}}{t}$) represents the magnitude and direction of an emotion's change of *velocity* within the emotion space over a unit of time

Definition 5. Mass represents an emotional state's tendency to maintain a constant velocity unless acted upon by an external force; quantitative measure of an emotional object's resistance to the change of its velocity

Definition 6. Force ($\mathbf{F} = m \cdot \mathbf{a}$) is an external influence that causes an emotional state to undergo a change in direction and/or velocity

Laws of Emotion Dynamics

The following are two laws that form the basis of emotion dynamics, to be used in order to explain and investigate the variance of emotional states within the emotional space. They describe the relationship between the forces acting on an emotional state and its motion due to those forces. They are analogous to Newton's first two laws of motion. The emotional states upon which these laws are applied (as in Newton's theory) are idealized as particles (the size of the body is considered small in relation to the distance involved and the deformation and rotation of the body are of no importance in the analysis).

Theorem 1. *The velocity of an emotional state remains constant unless it is acted upon by an external force.*

Theorem 2. *The acceleration* $\mathbf{a}$ *of a body is parallel and directly proportional to the net force* $\mathbf{F}$ *and inversely proportional to the mass m:* $\mathbf{F} = m \cdot \mathbf{a}$

Emotion Center and Gravity

The emotion space has a center, the agent's neutral state, a point in space to which all of the agent's emotional states tend to gravitate (usually (0,0,0,0), however, different characters might be predisposed to certain kinds of emotions, thus, we should be able to specify a different emotional centre, and instill a certain disposition, for each character; we also use different emotion state mass to show how easy/hard it is to change an agent's mood). In order to represent this tendency we use gravitational force:

$$\mathbf{G} = m \cdot \frac{\mathbf{p} - \mathbf{c}}{||\mathbf{p} - \mathbf{c}||} \cdot k_g,$$

where p is the current position, c is the center and k_g is a gravitational constant. This force ensures that, unless acted upon by other external forces, the emotional state will decay towards the emotion space center.

4 Emotion System Architecture

We developed an emotion subsystem architecture that interposes itself between the agent's behavior module and the environment. The subsystem models the process described by Lazarus (Sec. 2).

Events perceived from the environment are first processed by the appraisal module, where an emotional force is associated with it. The resulting list of events is then sorted in descending order according to magnitude and fed into the agent's perception module. This is done in accordance with the psychological theory of attention narrowing presented in Sec. 2, so that in a limited attention span scenario, the agent will be aware of the more meaningful events.

We treat the agent behavior module as a black box that may house any behavior generation technique (rule based expert system, behavior trees, etc.). The interface receives as input events perceived from the environment and produces a set of possible actions. The conflict set module takes this set of rules and evaluates them in order to attach an emotional state to each. Out of these, the action whose emotion vector most closely matches the agent's current state is selected to be carried out.

$$\underset{i \in \{conflictset\}}{\arg\min} \quad arccos \frac{e_{agent} \cdot e_i}{||e_{agent}|| \cdot ||e_i||}$$

Feedback from the environment has an associated emotion force vector that actually affects the agent's emotional state. Feedback is distributed to the appraisal and conflict set modules as well as the agent's behavior module. Actions undertaken are temporariliy stored in an action buffer for feedback purposes.

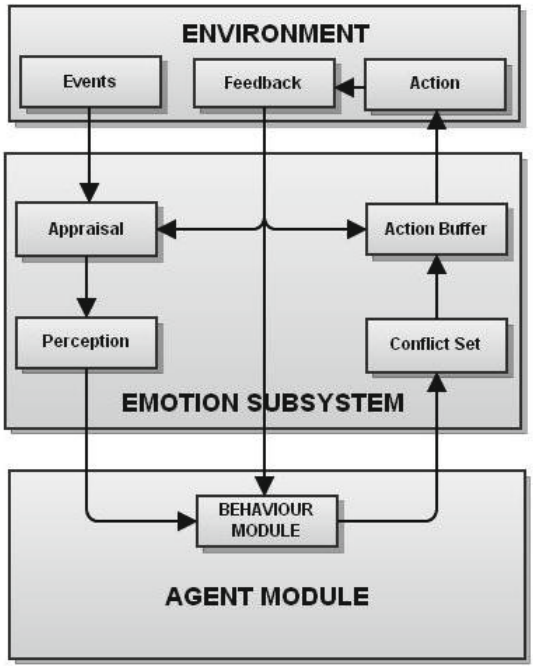


Fig. 3 Emotion system architecture

Learning

Both the appraisal and conflict set modules use machine learning techniques in order to learn to predict the outcome of events and actions, respectively. As there are many viable options for which learning technique to use (hidden markov models, neural nets, q-learning), we use a plug-and-play model where the technique can be replaced.

The appraisal module attempts to predict the emotional feedback received from the environment based on characteristics of the event to classify and previous feedback. The goal of the appraisal module is to better label (and thus establish priority of) events for the perception model. The conflict set module works in a similar way based on the characteristics of actions taken. The goal of the conflict set module is to settle conflicts between competing actions by selecting the most appropriate action to be performed. This is why an action is selected based on the lowest emotional distance to the agent's current state instead of magnitude.

Both modules treat the event, rule respectively, configuration as the observable state of the model, and attempt to predict the next hidden state (the emotional feedback force).

5 Conclusion and Future Work

We succeeded in developing a light-weight and scalable emotion representation and evaluation model to be used by virtual agents. The model is suitable for fast real-time evaluation and simulation such as that required in our chosen application type, computer role-playing games.

We also devised a plug and play emotion subsystem architecture that can be used with any agent behavior model and any machine learning technique. This allows us to provide artificial emotion simulation and modify agent behavior based on the given simulation. By using the model proposed in this paper, there is no need to adapt the game engine architecture to accomodate computationally expensive artificial intelligence techniques.

We chose computer role-playing games as an application domain due to personal preference, however, the system can be used in any type of application that requires emotion simulation. Moreover, other applications may not have the processing power restrictions that running along side CPU intensive special effects and physics simulations impose, and would leave more compuring power for the artificial intelligence module.

Collective Emotion

One interesting research direction is applying the model to multi-agent system techniques, such as swarm or ambient intelligence, in order to endow the entire system with collective emotions. Interacting agents would influence each other's moods based on the newtonian emotion model and have an emotion dissemination protocol over the multi agent system. The technology could further be integrated with ambient intelligence applications to either influence people's mood in order to make them more productive/happier or make them aware of the state of the ambient network around them in a non-invasive manner.

Acknowledgements. The work has been funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/88/1.5/S/61178.

References

1. Burke, A., Heuer, F., Reisberg, D.: Remembering emotional events. Memory & Cognition 20, 277–290 (1992)
2. de Sousa, R.: Emotion. Stanford Encyclopedia of Philosophy (September 2010), http://plato.stanford.edu/entries/emotion/
3. Kensinger, E.A.: Remembering emotional experiences: The contribution of valence and arousal. Reviews in the Neurosciences 15, 241–251 (2004)
4. Lazarus, R.S.: Emotion and Adaptation. Oxford University Press, New York (1991)

5. Plutchik, R.: Emotion: Theory, research, and experience. Theories of emotion, vol. 1. Academic, New York (1990)
6. Schachter, S., Singer, J.: Cognitive, social, and physiological determinants of emotional state. Psychological Review 69, 379–399 (1962)
7. Sharot, T., Phelps, E.A.: How arousal modulates memory: Disentangling the effects of attention and retention. Cognitive, Affective & Behavioral Neuroscience 4, 294–306 (2004)

Author Index