Lars Braubach
Wiebe van der Hoek
Paolo Petta
Alexander Pokahr (Eds.)

LNAI 5774

Multiagent System Technologies

7th German Conference, MATES 2009
Hamburg, Germany, September 2009
Proceedings

Lecture Notes in Artificial Intelligence 5774

Edited by R. Goebel, J. Siekmann, and W. Wahlster

Subseries of Lecture Notes in Computer Science

Lars Braubach Wiebe van der Hoek
Paolo Petta Alexander Pokahr (Eds.)

Multiagent System Technologies

7th German Conference, MATES 2009
Hamburg, Germany, September 9-11, 2009
Proceedings

Springer

Series Editors

Randy Goebel, University of Alberta, Edmonton, Canada
Jörg Siekmann, University of Saarland, Saarbrücken, Germany
Wolfgang Wahlster, DFKI and University of Saarland, Saarbrücken, Germany

Volume Editors

Lars Braubach
Alexander Pokahr
University of Hamburg
Distributed Systems and Information Systems
Hamburg, Germany
E-mail: {braubach, pokahr}@informatik.uni-hamburg.de

Wiebe van der Hoek
University of Liverpool
Agent Applications, Research, and Technology (Agent ART)
Liverpool, UK
E-mail: Wiebe.Van-Der-Hoek@liverpool.ac.uk

Paolo Petta
Austrian Research Institute for Artificial Intelligence (OFAI)
Intelligent Software Agents and New Media Group
Vienna, Austria
E-mail: paolo.petta@ofai.at

Library of Congress Control Number: 2009933194

CR Subject Classification (1998): I.2, I.2.11, C.1.4, C.2.1, C.2.4, D.2, D.3, D.1.3

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN 0302-9743
ISBN-10 3-642-04142-6 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-04142-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12753154 06/3180 5 4 3 2 1 0

Preface

The seventh German Conference on Multi-Agent System Technologies (MATES 2009) took place in Hamburg, Germany, during September 9–11, 2009, and was colocated with the tenth International Workshop on Computational Logic in Multi-Agent Systems (CLIMA X) and the Fifth International Workshop on Modelling of Objects, Components, and Agents (MOCA 2009). MATES 2009 was organized by the German "Gesellschaft für Informatik" (GI) Special Interest Groups on "Distributed Artificial Intelligence" and "Communication and Distributed Systems" in cooperation with the Steering Committee of MATES.

One main aim of the MATES conference series consists in bringing together researchers from around the world and providing a fruitful discussion basis for exchanging ideas and sharing the newest scientific results. Since its inception in 2003, MATES has been colocated with mainstream software engineering conferences like the Net.ObjectDays as well as with the German Artificial Intelligence Conference (KI) and has thus strived to address the full range of agent research topics ranging from practical applications and tools for agent technology to the theoretical foundations of multi-agent systems. In addition to the broad range of topics covered by MATES, in recent years special areas of interest (hot topics) within the field of multi-agent systems have also been identified and have influenced the conference editions. In 2009 the focus was laid on complex, adaptive systems and phenomena of self-organization.

The international Program Committee for MATES 2009 carefully selected among 44 submissions from all over the world 14 to be presented as full papers, 10 to be presented as short papers and finally also five exhibition papers. Besides the regular paper presentations, the exhibition papers also included a live software demonstration at the conference venue. Furthermore, the program featured an invited talk by the distinguished speaker Birgit Burmeister (Daimler AG) as well as a doctoral colloquium and mentoring event.

Finally, as Chairs, and in the name of all members of the Steering Committee, we would like to thank all authors of submitted papers and the invited speaker for their contributions, all members of the Program Committee as well as additional reviewers for their careful, critical and constructive reviews, and the local conference organizers including student assistants as well as all others involved in helping to make MATES 2009 a success.

July 2009

Lars Braubach
Paolo Petta
Wiebe van der Hoek
Alexander Pokahr
Winfried Lamersdorf

Organization

General Chair

Winfried Lamersdorf	Universität Hamburg, Germany

Program Co-chairs

Lars Braubach	Universität Hamburg, Germany
Wiebe van der Hoek	University of Liverpool, UK
Paolo Petta	OFAI, Austria
Alexander Pokahr	Universität Hamburg, Germany

Doctoral Consortium Chair

Ingo J. Timm	Goethe-Universität Frankfurt, Germany

Steering Committee

Hans-Dieter Burkhard	Humboldt-Universität zu Berlin, Germany
Stefan Kirn	Universität Hohenheim, Germany
Matthias Klusch	DFKI, Germany
Jörg P. Müller	Technische Universität Clausthal, Germany
Rainer Unland	Universität Duisburg-Essen, Germany
Gerhard Weiss	SCCH Hagenberg, Austria

Program Committee

Klaus-Dieter Althoff	Universität Hildesheim, Germany
Elisabeth André	Universität Augsburg, Germany
Bernhard Bauer	Universität Augsburg, Germany
Federico Bergenti	Università degli Studi di Parma, Italy
Michael Berger	Docuware AG, Germany
Ralph Bergmann	Universität Trier, Germany
Rafael Bordini	Federal University of Rio Grande do Sul, Brazil
Hans-Dieter Burkhard	Humboldt-Universität zu Berlin, Germany
Keith Clark	Imperial College, UK
Mehdi Dastani	Universiteit Utrecht, The Netherlands
Jörg Denzinger	University of Calgary, Canada
Jürgen Dix	Technische Universität Clausthal, Germany
Torsten Eymann	Universität Bayreuth, Germany
Klaus Fischer	DFKI, Germany

Maria Ganzha	Elblag University of Humanities and Economy, Poland
Mike Georgeff	Monash University, Australia
Paolo Giorgini	Università degli Studi di Trento, Italy
Jos van Hillegersberg	Universiteit Twente, The Netherlands
Benjamin Hirsch	Technische Universität Berlin, Germany
Michael Huhns	University of South Carolina, USA
Catholijn Jonker	Delft University of Technology, The Netherlands
Stefan Kirn	Universität Hohenheim, Germany
Franziska Klügl	Örebro University, Sweden
Matthias Klusch	DFKI, Germany
Daniel Kudenko	University of York, UK
Gabriela Lindemann	Humboldt Universität zu Berlin, Germany
Stefano Lodi	Università di Bologna, Italy
Beatriz Lopez	Universitat de Girona, Spain
Viviana Mascardi	Università degli Studi di Genova, Italy
Mirjam Minor	Universität Trier, Germany
Heinz-Jürgen Müller	Duale Hochschule Baden-Württemberg Mannheim, Germany
Daniel Moldt	Universität Hamburg, Germany
Jörg P. Müller	Technische Universität Clausthal, Germany
Andrea Omicini	Università di Bologna, Italy
Marcin Paprzycki	Polish Academy of Sciences, Poland
Adrian Paschke	Freie Universität Berlin, Germany
Michal Pechoucek	Czech Technical University in Prague, Czech Republic
Wolfgang Renz	Hochschule für Angewandte Wissenschaften Hamburg, Germany
Alessandro Ricci	Università di Bologna, Italy
Abdel Badeh Salem	Ain Shams University, Egypt
Von-Wun Soo	National Tsing-hua University, Taiwan
Amal El Fallah Seghrouchni	University Pierre and Marie Curie, France
Ingo J. Timm	Goethe-Universität Frankfurt, Germany
Adelinde Uhrmacher	Universität Rostock, Germany
Rainer Unland	Universität Duisburg-Essen, Germany
Laszlo Zsolt Varga	MTA SZTAKI, Hungary
Danny Weyns	Katholieke Universiteit Leuven, The Netherlands
Cees Witteveen	Delft University of Technology, The Netherlands
Georg Weichhart	Johannes Kepler Universität Linz, Austria
Gerhard Weiss	SCCH Hagenberg, Austria

Additional Reviewers

Rashad Badawy
Tina Balke
Tristan Behrens
Petr Benda
Lawrence Cabac
Michael Duvigneau
Jiri Hodik
Stefan König
Régis Newo
Christoph Niemann
M. Birna van Riemsdijk
Daniel Schmalen
Peter Schuur
Michal Sindlar
Bas Steunebrink
Matthias Wester-Ebbinghaus

Table of Contents

Invited Talk

Full Papers

Short Papers

Exhibition Papers

Industrial Application of Agent Systems: Lessons Learned and Future Challenges

(Extended Abstract)

Birgit Burmeister

Daimler AG, Group Research & MBC Development, H515, GR/PPF
Leibnizstraße 2, 71032 Böblingen, Germany
birgit.burmeister@daimler.com

Challenges in modern industry such as high complexity inside companies and in the environment, ever increasing dynamics to react to, integration of many different partners and world-wide distribution of supply and production networks seem to really demand for the promises of agent systems. Multi agent systems handle complex systems by dividing them into smaller pieces. Agents can react to changing environments. Distribution, autonomy and interaction are basic concepts at the heart of agent systems.

Nevertheless applications of agent systems in industry are still the exception rather than the normal case. This contribution will describe some applications of agent systems which were developed at Daimler Research during the last ten years. We will present the advantages and lessons learned from these cases. But we will also have a look at the obstacles in applying agent systems in a large scale. As a result we will sketch some resulting challenges from the point of view of industry for the (scientific) agent community.

An important application of agent systems in Daimler AG was developed during a project called Production 2000+ addressing the challenges of manufacturing in modern automotive industry. The system implemented a market-based approach to provide a flexible flow of workpieces through a manufacturing facility for cylinder heads [1], [2], [3].

The agent system was used to control a pilot facility in Stuttgart-Untertürkheim. The facility ran from 2000 to 2003 for operational production. Although technically very successful the pilot was not distributed into a larger environment for two reasons: Firstly, the multi-purpose machines needed for enabling the flexible control realised by the agent system are much more expensive than “normal” single use machines, i.e. the cost for a potentially necessary flexibility to handle disturbances and failures are higher than the cost for a normal, not very flexible line. The costs that occur later when disturbances and failures happen are most often not considered. Secondly, many components of up-to-date manufacturing equipment, like e.g. equipment for transportation, nowadays already offer a certain extent of flexibility, without the need for an agent based control systems, although some ideas from agents are also used here, without being named “agents” [4], [5].

Although inspired by the ideas of Production 2000+ concerning flexibility in 2005 we started working in a quite different area of applying agent technology, namely that

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 1–3, 2009.

of business process management [6], [7], [8]. Business processes in today's companies are highly complex, involve many different participants and spawn multiple information systems. Running business processes is no longer possible without support by information technology. Moreover, optimizing business processes is crucial for business success of companies. Therefore, the processes have to be continuously improved and have to be flexible enough to deal with the dynamic environment. Tools used nowadays in this area support a very simple mind-model behind modelling: processes are seen as long and fixed sequences of activities, which is far away from reality and from the challenges. This leads to the fact that models drawn in such tools are often used only to cover white walls in the offices. The processes really executed in the companies are different from that on the wallpapers, "shadow" processes dominate the "official" ones and IT systems are not understood or inflexible and hence misused by many users.

Therefore we have developed a new modelling and execution approach based on agent technology. Our modelling approach defines a process model with goals, contexts, and plans using the basic concepts of the BDI agent architecture. The execution semantics of the BDI architecture is used to execute a goal-based process model. This leads to so called process agility, both in process execution and in process and system implementation. The approach was demonstrated in the area of the engineering change management process. The next generation of engineering change management at Mercedes-Benz Cars will be based on this approach. The system is currently under implementation and will be rolled out for a first pilot in 2010.

In future we would like to come back to the ideas on Production 2000+. The ideas of distributed decision making and control and self-organization can be used to tackle current challenges in production and logistics. Combined with technical progress in the area of "auto-ID" technology and especially RFID (radio frequency identification) technology we see a revival of agent techniques. The auto-ID technology can be used to provide a seamless information flow which can be used to support the distributed and autonomous decision making. Moreover with increasing computing power and decreasing prices of RFID-tags also really distributed agents can be implemented into these tags and thus closely coupled to the agents' hardware, like e.g. workpieces, machines, transportation units etc.

To get the full advantages of the agent approach some work needs to be done: One of the most important promises of agent systems is the enabling and support for flexibility. For any application case it is crucial to determine and to not exceed the actually needed flexibility (as it was the case with the Production 2000+ approach.) For these requirements a technically appropriate agent system should be defined, that should be also as simple as possible. Especially for applications in production and logistics an engineering approach and methodology is needed, which defines which agent architecture to build, which communication mechanisms to use etc.

Another important aspect is the cost-benefit analysis. For any IT application in industry the benefits must be qualified, and even better quantified. Since agent system offers solutions, which cannot (or not easily) be implemented with other approaches to IT-Systems it is not easy to compare agent systems to these other approaches and especially quantify the advantages. Therefore different kinds of measures are needed for the advantages of agents, such as flexibility, decentralisation etc. This will help to

really support an informed decision making about the question if the implementation of an agent system pays off.

References

1. Bussmann, S., Schild, K.: Self-organising manufacturing control: an industrial application of agent technology. In: Proc. of the 4th Int. Conf. on Multi-Agent Systems (ICMAS 2000), Boston, USA, pp. 87–94 (2000)
2. Bussmann, S., Schild, K.: An Agent-Based Approach to the Control of Flexible Production Systems. In: Proc. of the 8th IEEE Int. Conf. on Emergent Technologies and Factory Automation (ETFA 2001), Antibes Juan-les-pins, France, vol. 2, pp. 481–488 (2001)
3. Sundermeyer, K., Bussmann, S.: Einführung der Agententechnologie in einem produzierenden Unternehmen - Ein Erfahrungsbericht. Wirtschaftsinformatik 43(2), 135–142 (2001) (in German)
4. Schild, K., Bussmann, S.: Self-organization in manufacturing operations. Communications of the ACM 50(12), 74–79 (2007)
5. Bussmann, S.: Production 2000+: Implementing and deploying an agent-based control system, `http://www.stefan-bussmann.de/index.php?pg=P2pLessons` (last visited 24.6.2009)
6. Burmeister, B., Steiert, H.-P., Bauer, T., Baumgärtel, H.: Agile Processes through Goal- and Context-oriented Business Process Modeling. In: Eder, J., Dustdar, S. (eds.) BPM Workshops 2006. LNCS, vol. 4103, pp. 217–228. Springer, Heidelberg (2006)
7. Rimassa, G., Burmeister, B.: Achieving Business Process Agility in Engineering Change Management with Agent Technology. In: Workshop dagli Oggetti agli Agenti WOA 2007 - Agents and Industry: Technological Applications of Software Agents, Genua, September 2007, pp. 1–7 (2007)
8. Burmeister, B., Arnold, M., Copaciu, F., Rimassa, G.: BDI-Agents for Agile Goal-Oriented Business Processes. In: Berger, M., Burg, B., Nishiyama, S. (eds.) Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008) – Industry and Applications Track, Estoril, Portugal, pp. 37–44 (2008)

Multi-Agent Navigation Using Path-Based Vector Fields

Tristan Behrens, Randolf Schärfig, and Tim Winkler

Department of Informatics, Clausthal University of Technology,
Julius-Albert-Straße 4, 38678 Clausthal, Germany
{tristan.behrens,randolf.schaerfig,tim.winkler}@tu-clausthal.de

Abstract. We present an approach to multi-agent navigation, that is based on generating potential-fields from A*-paths. We will introduce and compare two algorithms: 1) a geometrical algorithm that is based on quads, and 2) an images-based algorithm. We show empirically that the images-based algorithm is more memory-consuming, but has better performance.

1 Introduction

Multi-agent systems programming is a promising software engineering paradigm [3,4]. It is especially suited for the development of systems that have to operate in dynamic environments [7].

In [2], we have introduced an approach to agent-oriented control of entities in a simulated, physical environment. We have introduced an agent-entity-relation: it allows agents to control entities, whereas each agent can control a set of entities. This approach makes sense especially when several entities have the same goal(s).

Furthermore we let the agents steer their associated entities using the potential fields method [1,13,15,11]. Each agent can control its entities by putting attractive and repelling forces into the environment. The entities are affected by these forces – they behave like particles in a force field – and move accordingly.

In this paper we present a combination of A* path-planning and the potential fields method. We 1. motivate our research, 2. introduce two computer graphics approaches to generating potential fields from A^* paths, and 3. compare the two approaches.

In section 2, we present the problem we would like to solve and propose a multi-agents systems solution. In section 3 we lay the groundwork for steering entities using the potential fields method. In section 4, we elaborate on and compare the two algorithms that we have developed to generate potential fields from given A^*-paths. In section 5 we present some related work. Finally, we conclude and give an outlook to future work.

2 MAS

Firstly, we will give a brief description of the problem we would like to solve. Let a physical environment be given. Some areas of the environment are accessible,

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 4–15, 2009.

some are not. Situated in that environment is a set of entities (e.g. robots). Furthermore, we have a set of agents, that can control these entities. There are several questions that should be answered:

- How can the agents map the environment?
- How can the agents find out which areas are accessible and which ones are not?
- How can agents steer the entities from one place to another?

In our approach we use two kinds of agents: *entity agents* and *mapper agents*. Entity agents control the entities in the environment. Each one of these agents is associated with several entities. A single mapper agent is responsible for mapping the environment and providing the entity agents with paths if requested. Also the mapper agent can provide entity agents with paths to unknown areas in the environment if requested.

These agents should cooperate to solve the given problem. Our solution is as follows. The mapper agent internally represents the environment in a discretized fashion by a grid. Each cell of the grid can be either *accessible*, *blocked*, or *unknown*. The agent is not embodied (i.e. it cannot act and perceive in the environment) and thus has to rely on other agents to provide it with informations about the environment. The size and resolution of the grid can be defined by the agent itself.

Each entity agent is indirectly embodied in the environment. It can access all the sensors of its associated entities and thus is capable of perceiving what the entities can perceive. Furthermore it can access the associated entities' actuators, allowing it to act in the environment.

All agents are capable of exchanging informations via message-passing. Entity agents can request paths through the environment from the mapper agent. The latter takes into account its internal environment representation, invokes the A* algorithm to compute a shortest path, and returns the path back to the respective entity agent.

3 Navigation

The potential fields method solves a navigational problem by representing the topology and structure of the environment as a combination of attracting and repelling forces. A potential field in general is a function $f : \mathbb{R}^2 \to \mathbb{R}^2$ that maps a two-dimensional vector to another one. Usually the input-vector represents a position on the Euclidean plane (in our case in the environment) and the output-vector a force that is effective at that position. A moving entity follows the force vector that affects it at its position.

Example 1 (potential fields). Any function

$$f_{gauss} : [x, y] \mapsto \frac{[x - x_0, y - y_0]}{|| \, [x - x_0, y - y_0] \, ||} \cdot a \cdot \exp\left(-\frac{|| \, [x - x_0, y - y_0] \, ||^2}{2s^2}\right)$$

is called a *Gaussian repeller* potential field. The constant vector $[x_0, y_0]$ represents the *center*, and the constants a and s represent the *amplitude* and the *spread* of the field respectively. The repelling force is strongest at the center and steeply falls off, converging to 0. An entity approaching a Gaussian repeller will be affected once it gets close to that force. The amplitude a determines the maximum strength of the force. The spread s determines the width of the Gaussian bell and thereby the range of influence of the field.

Another potential field is the sink attractor:

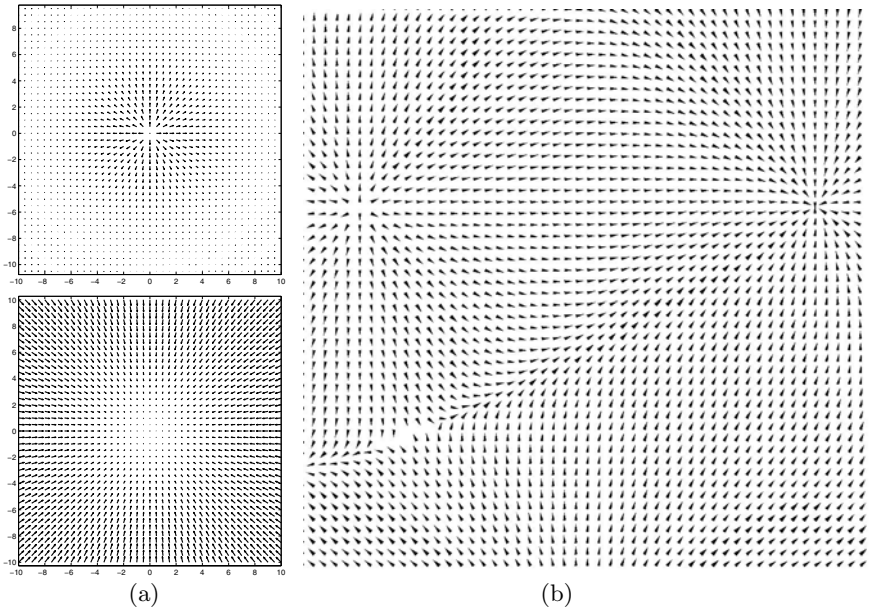
$$f_{sink} : [x, y] \mapsto \frac{[x - x_0, y - y_0]}{|| [x - x_0, y - y_0] ||} \cdot g_{sink}(x, y)$$

with

$$g_{sink} : [x, y] \mapsto a \cdot \exp\left(-\frac{|| [x - x_0, y - y_0] ||^2}{2s^2}\right) - g \cdot || [x - x_0, y - y_0] || - a$$

The constant vector $[x_0, y_0]$ represents the *center*. The constants a and s represent the *amplitude* and the *spread* respectively. The constant g represents the grade. In the sink attractor the attractive force is stronger the farther away the target is. It is the combination of a conical potential field and a Gaussian one.

Given a set of potential fields, the overall potential field is simply the sum of all potential fields that are relevant. Figure 1a shows a potential field that is the



(a) (b)

Fig. 1. Examples for the potential-fields method. The left side shows the Gaussian repeller and the sink attractos. The right side shows the sum of two Gaussian repellers (top left and bottom) and a sink attractor (top right).

sum of two Gaussian repellers (top left and bottom) and a sink attractor (top right) represented as a vector field. An entity that starts at a position in the middle would move away from the repellers and move towards the sink.

The method is very elegant. It is easy to model environmental characteristics with potential fields. Also, it is extremely handy when dealing with several moving entities. As an AI programmer you do not have to deal with movement on a microscopic level for each entity. Instead you just let them all follow the shared potential field like particles in a stream.

Although the method is elegant and allows entity movement in a unified way, there are also drawbacks. The worst drawback is the fact that entities could get stuck if there are local optima in the potential field. Thus an improvement is necessary. Moving a stuck entity a bit might release it from the local optimum, but this method is not useful if the potential field is too complex. Path-planning does improve the situation significantly.

4 Path Based Potential Fields

Our approach is a compromise approach, that, although it does not rule out getting stuck in local optima, reduces the probability of it. We distinguish two classes of obstacles: *static* and *dynamic* ones. Static ones are avoided by path-planning using the well-known A* algorithm, dynamic ones are avoided by repelling potential fields, both in an unified framework.

The A* algorithm is an informed search algorithm for finding shortest paths in graphs [16]. In our approach we use an implicit representation of the search graph. The environment is discretized as a grid. Each cell is a node in the graph if it is reachable. Two nodes are adjacent if the respective cells share a vertex or an edge.

We will now introduce and compare two algorithms for computing vector fields from arbitrary paths, that have been generated by the A* algorithm. The first algorithm is geometry-based, the second is raster-graphics based. The question we will answer is: given a path $P := (p_0, p_1, \ldots, p_n)$ with waypoints $p_i := [x_i, y_i] \in \mathbb{R}^2$, how can a potential field be calculated that lets the entities follow that path?

4.1 Geometry Based Approach

In the preprocessing phase, given a path $P := (p_0, p_1, \ldots, p_n)$, we generate a force-quad strip, which consists of one force-quad for each path segment $\langle p_i, p_{i+1} \rangle$. A force-quad is a tuple $q := \langle ls, le, ll, lr, d \rangle$, where $ls, le, ll, lr \subset \mathbb{R}^2$ are the lines defining the shape of the quad, and $d \in \mathbb{R}^2$ is the direction of its force. A force-quad strip is a sequence $Q := (q_0, q_1, \ldots, q_m)$, with $m = n - 1$ and where for each pair q_i, q_{i+1} the following holds: $le_i = ls_{i+1}$. Figure 2a shows an exemplary segment of a force-quad strip.

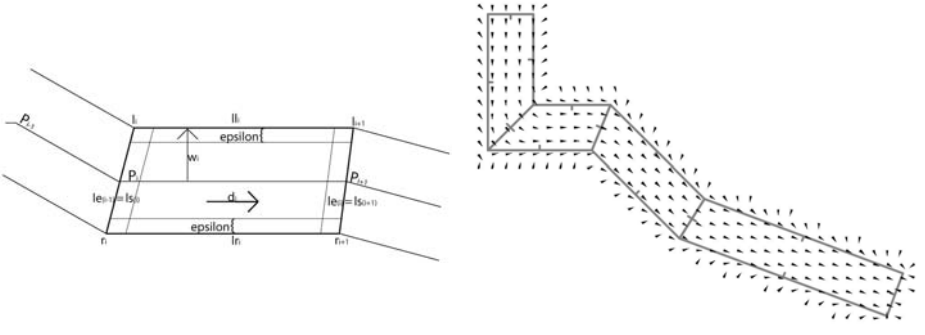
Calculating the force vector $f \in \mathbb{R}^2$ at a given environment position $p \in \mathbb{R}^2$ with respect to a given force-quad strip Q boils down to 1) detecting which force-quad $q_i := \langle ls_i, le_i, ll_i, lr_i, d_i \rangle \in Q$ contains p and 2) calculating f depending on d_i.

Preprocessing: Input-parameters for the force-quad strip generation are the path P and a width $w \in \mathbb{R}^+$. Algorithm 1 creates a sequence of connected force-quads. All quads have the same width and their ordering reflects the ordering of the waypoints of the path. Firstly, the direction vector $d_i := p_{i+1} - p_i$ is calculated for each path segment $\langle p_i, p_{i+1} \rangle$. The vector w_i is orthogonal to d_i and has the length w. The line ll_i is $\langle p_i, p_{i+1} \rangle$ translated by w_i and the line lr_i is $\langle p_i, p_{i+1} \rangle$ translated by $-w_i$. The algorithm then calculates the intersections of ll_i with ll_{i-1} and ll_{i+1} respectively, and the intersections of lr_i with lr_{i-1} and lr_{i+1} respectively. These four intersections define the force-quad

The first and the last quad are both special cases:

1. for the first quad the lines l_{-1} and r_{-1} are both set to $p_0 + -w_0 - d_0$, and
2. for the last quad the lines l_n and r_n are both set to $p_n + -w_n - d_n$.

This also allows handling the special case that the path consists only of two waypoints. Figure 2a shows a quad within a given path and all lines and vectors that the algorithm uses.



(a) (b)

Fig. 2. Illustrations for the force-quads based approach. The left side shows the lines, vectors, and values that the preprocessing algorithm uses. The right side shows a plot of the force field.

Calculating Force Vectors: We now consider the algorithm that computes the force-vector f for a given position $p \in \mathbb{R}^2$ with respect to a force-quad strip Q. The algorithm iterates over all the force-quads $q_i \in Q$ and checks if p is contained by q_i. If so, the algorithm computes the force-vector with respect to that quad. If p is not contained by any of the quads, the null vector is returned.

To get a smoother movement, the algorithm interpolates d_i of the force-quad q_i that contains p with respect to the position of p in q_i. It determines if the distance $dist$ of p to one of the borders of the quad is smaller than a given ϵ. If that is the case, the algorithm uses the linear interpolation of d and the normal vector n to calculate the resulting vector f. This is described by the following formula:

$$f = (dist/\epsilon) * d + (1 - (dist/\epsilon)) * n \tag{1}$$

Algorithm 2 computes the force-vector. Figure 2b shows a vector-field generated using the computed force-quad strip.

Algorithm 1. Preprocessing: generating a force-quad strip

Input: a path $P := (p_0, p_1, \ldots, p_n)$ and a width $w \in \mathbb{R}^+$
Output: a force-quad strip $Q := (q_0, q_1, \ldots, q_m)$ with $m := n - 1$

```
Q := ∅;
for all i ∈ [0, n − 1] do
    d_i = Normalize(p_i − p_{i+1})
    ll_i is (p_i, p_{i+1} translated by w_i; lr_i is (p_i, p_{i+1} translated by −w_i;
end for
if only two waypoints then
    for all i ∈ [0, n − 1] do
        if i = 0 then
            l_i = p_0 + w_i − d_i; r_i = p_0 − w_i − d_i
        else
            l_i = Intersection(ll_{i−1}, ll_i); r_i = Intersection(lr_{i−1}, lr_i)
        end if
        if i = n − 1 then
            l_{i+1} = p_n + w_i − d_i; r_{i+1} = p_n − w_i − d_i
        else
            l_{i+1} = Intersection(ll_i, ll_{i+1}); r_{i+1} = Intersection(lr_i, lr_{i+1})
        end if
        ls_i = line(l_i, r_i); le_i = line(l_{i+1}, r_{i+1})
        Save Quad ⟨ls_i, le_i, ll_i, lr_i, d_i⟩ in Q
    end for
else
    l_0 = Intersection(ll_0, (p_0 − d_0 + w_0)) r_0 = Intersection(lr_0, (p_0 − d_0 − w_0))
    l_1 = Intersection(ll_0, (p_1 + d_0 + w_0)) r_1 = Intersection(lr_0, (p_1 + d_0 − w_0))
    ls_0 = line(l_0, r_0); le_0 = line(l_1, r_1)
    Save Quad ⟨ls_0, le_0, ll_0, lr_0, d_0⟩ in Q
end if
return Q
```

Algorithm 2. Calculating Force Vectors

Input: a position vector $p \in \mathbb{R}^2$ and a force-quad strip $Q := (q_0, q_1, \ldots, q_m)$ threshold $\epsilon \in \mathcal{R}^+$
Output: a force vector $f \in \mathbb{R}^2$

```
for all q ∈ Q do
    if p is contained by q then
        f := interpolate(p, q, ε)
        return f
    end if
end for
return [0, 0]
```

4.2 Image Processing Approach

In this section, we describe a method that generates a vector-field from a given path $P := (p_0, p_1, \ldots, p_n)$, borrowing some machinery from image processing.

Preprocessing: Given a path P, the algorithm firstly creates a local frame $F \subset \mathbb{R}^2$, which is basically the enlarged bounding box of P. In the next step it translates one of the corners of F into the origin by subtracting the vector $o \in \mathbb{R}^2$ (see Fig. 3). The same is done for each waypoint.

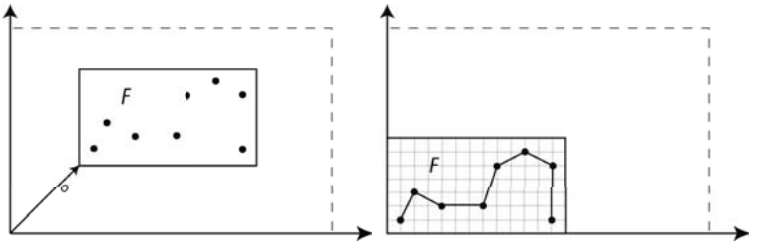


Fig. 3. The local frame F. The left side shows its alignment in the environment and the waypoints. The right side shows F after translation by o and plotting the path.

Since the goal is to create a discrete vector field with appropriate resolution, the coordinate axes are scaled by some factors $r_x, r_y \in \mathbb{Z}$ which are determined by the agent. Since it can prescribe the desired resolution of F.

This yields a discrete representation of the local frame F, which is interpreted from now on as a 2D image. The image is then initialized with an intensity (color) value. After that, the waypoints are connected by an arbitrary discrete line drawing algorithm with an certain line-width w depending on the use case. A thin line would for example be painted if the entities are supposed to move in single file and a thick line if they should maintain their formation. Also the path is rendered with a gradient starting with a high intensity and ending with a low intensity. This will provide the "current" of the "river" (see Fig. 4). Finally the image is smoothened in order to ensure smooth movement by applying a Gaussian operator. The continuous Gaussian function is

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}}. \tag{2}$$

Since the image is already a raster of discrete pixels, we also need a discrete version of the Gaussian operator. Two common approximations depending on the value of σ in equation (2) are:

$$G(x, y) = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{3}$$

$$G(x, y) = \frac{1}{423} \begin{pmatrix} 2 & 7 & 12 & 7 & 2 \\ 7 & 31 & 52 & 31 & 7 \\ 15 & 52 & 127 & 52 & 15 \\ 7 & 31 & 52 & 31 & 7 \\ 2 & 7 & 12 & 7 & 2 \end{pmatrix} \tag{4}$$

which directly reflects the property of σ as smoothing parameter. The bigger the smoothing parameter, the smoother we expect the movement to be. We apply a discrete convolution (5) to the image by either using (3) or (4) as the filter kernel and get the desired result (e.g. see [8], [17]). The pseudo code for this preprocessing step is given in Algorithm 3. Figure 4 shows a painted path after applying two blur-operators with different strength.

$$(I * G)(x, y) = \sum_i \sum_j I(i, j)G(x - i, y - i) \tag{5}$$

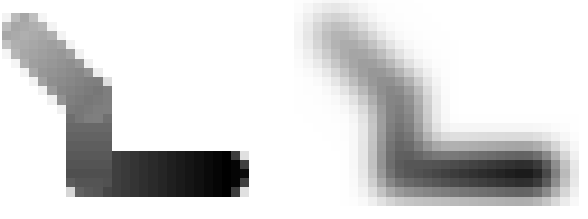


Fig. 4. The same path after applying two Gaussian operations with different intensities. The higher the intensity the smoother the movement.

Algorithm 3. Preprocessing

Input: a path $P := (p_0, p_1, \ldots, p_n)$ a scaling factor r, the line-width w, and the blur intensity b

Output: $< F, T >$ the local frame F and the transformation T

$B := calcBoundingBox(P)$
$F := initImage(B.width/r, B.height, /r, highestIntensity)$
$T := getTransformation(B, r)$ //maps from environment to frame coordinates
for all p_i **do**
  $p_i := T(p_i)$ //map all waypoints to frame-coordinates
end for
$F := drawPolyLine(F, P, highIntensity, lowIntensity, w/r)$
$F := blur(F, b)$
return $< F, T >$

Calculating Force Vectors: Given a query point $q \in \mathbb{R}^2$ the algorithm performs the following steps to retrieve the respective force-vector: 1) transforming the environment position p into the respective position p' in the frame by translating and scaling p respectively, 2) calculating the intensity difference between p' and all of its eight direct neighbors, and 3) retrieving the force with respect to the difference of intensities. The pseudo code for the retrieval is given in Algorithm 4.

Algorithm 4. Calculating Force Vectors

Input: a position vector $p \in \mathbb{R}^2$, a local frame F and a transformation T
Output: a force vector $f \in \mathbb{R}^2$

```
q := T(q) //map to frame coordinates
c_q := getColorAt(F, q)
prevDiff := 0; m := 0; n := 0
for i := −1 to 1 do
  for j := −1 to 1 do
    diff := c_q − F(q[x] + i, q[y] + j)
    if diff > prevDiff then
      m = i; n = j; prevDiff = diff
    end if
  end for
end for
return (m, n) ∗ prevDiff
```

4.3 Implementation and Comparison

We base our experiments on a computer-game-like physical world. Many computer games make the cooperation of entities (optionally with a human player) desirable. Modern computer games almost always feature a very detailed simulation of virtual environments. It is straightforward to create sensors and actuators for agents that are situated in a game world. And you do not have to deal with hardware. Our experimental game world is an 2APL-environment. 2APL[6] is

# Nodes	s Quads	s Raster
21	5.9	5.8
19	5.6	5.8
17	5.5	5.8
15	5.1	5.8
13	4.6	5.8
11	4.2	5.7
9	3.9	5.8
7	3.1	5.8
5	2.3	5.6
3	2.4	5.4

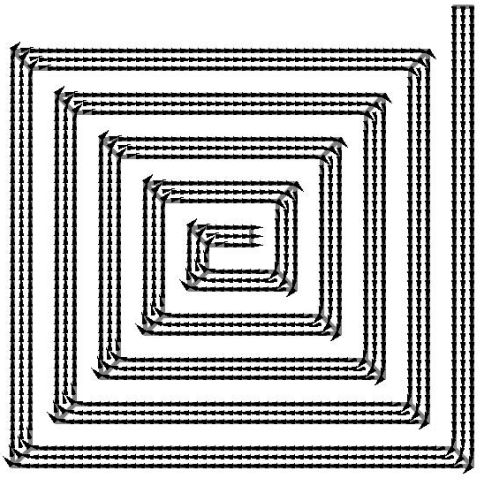


Fig. 5. Comparison of the two algorithms. The experiments were done on a 2 GHz Intel Core Duo MacBook Pro with 2GB RAM. The left side show the results. The right side shows the path that has been used.

an agent-programming platform that is based on the BDI-methodology. Details about the environment and the implementation can be found in [2].

To compare the two algorithms we have set up a simple test environment. We begin the test with an 21 node path (see Fig. 5). We then successively remove the last waypoint until we end up with a two-node path. For each path we generate two potential fields. One is generated using the quad-based algorithm, the other is created with the raster-based one. Each field is then queried a million times with random positions and the time needed is measured.

As you can see from the table in Fig. 5 the quads-based algorithm performs worse for long paths. We have expected this since for each additional node around four more dot-producs have to be computed. The raster-based algorithm almost always takes the same time. From the implementation point of view the pre-processing step in the raster-based approach is more straightforward since it boils down to using the computer graphics API. For short paths the raster-based approach is more memory consuming. The quad-based algorithm needs approximately $(n-1)\cdot 14$ doubles space given a path-length n. The raster-based algorithm's memory consumption depends on the area covered by the path and the resolution. Finally from the user experience-point of view, moving using the quads-based algorithm is more convincing. It yields a very smooth movement of the entities.

5 Related Work

In our recent work, we have combined A* with potential fields in a more primitive fashion. Instead of generating a potential field from a given path directly, we have associated each node of the path with a sink attractor. A force has been activated if the respective node has been the next to be reached and it has been deactivated once the node has been reached. Although the approach worked very good, the movement was not convincing because it lacked smoothness.

We have also concentrated on multi-agent systems dynamics that are useful for implementing computer games AI. The high dynamics of game worlds with respect to the number of entities (entities can be created and removed from the environment) and the distribution of different task among the entities made it necessary to allow respective dynamics in the MAS as well. A run of a MAS usually begins with an environment in its initial state (the topology of the map and the positions of the initial set of entities) and one default agent. We then allow agents to instantiate more agents if necessary, and we allow them to transfer their responsibility over associated entities to other agents. This can be exploited for example to split or merge groups of entities.

Hagelbäck et al. [10,9] did similar research in this area. They use a discretization for the complete algorithm, which represents the potential field. Our approaches differ especially in the structure of the MASs. They have for each entity one controlling *unit agent*, and another agent that coordinates those unit agents. We do not consider entities to be agents.

In their paper[12], Koren and Borenstein discuss problems inherent to the potential fields method. They identify four significant problems: trap situations

due to local optima, no passage between closely spaced obstacles, oscillations in the presence of obstacles, and oscillation in narrow passages. Mamei and Zambonelli[14] apply potential fields to the computer game Quake 3 Arena. In their article[18], Weyns et al. solve the problem of task-assignment in MAS inter alia with a field-based approach. This is, tasks emit fields in the environment that attract idle agents.

6 Conclusions and Future Work

In this paper we have shown how to combine A* path-finding and potential fields in a multi-agent setting. We have proposed a MAS with separated responsibilities. One agent is responsible for mapping the environment and calculating shortest paths, other agents are responsible for steering sets of entities. We have introduced and compared two algorithms for generating potential fields from A^*-paths. The first one is geometry based and relies on the generation of quads that model the path. Calculating the force-vector at a given position boils down to checking in which quad contains the position. The second algorithm is raster-graphics based. It plots a path into an image and blurs the latter to allow smoother movement. Calculating the force-vector is done by finding the pixel that is underneath the position and calculating the direction to the neighboring pixel that decreases the height. Our experiments have shown, that the raster-graphics approach is faster when dealing with long paths. For paths with a smaller number of nodes, the geometry algorithm is more efficient.

We believe that the two algorithms for generating potential-fields from A^* paths could be optimized. First of all, the geometry-based one could be improved by using a more elaborate preprocessing step. We think about using for example the binary-space partitioning technique to increase the speed of calculating force vectors. The rastered algorithm on the other hand could be improved when it comes to memory-comsumption. Only a necessary area of the heightmap could be rendered.

We intend to shift away from our custom-made 2APL-environment towards ORTS [5]. ORTS (Open Real-Time Strategy) has been established for studying AI problems in a real-time setting. This would allow for a more elaborate problem description of a more realistic problem setting. Also we would like to concentrate more on the actual structure of the MAS when it comes to the types and the numbers of agents, since our proposal in this paper is superficial but sufficient for our purposes.

References

1. Arkin, R.C.: Behavior-Based Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, Cambridge (1998)
2. Behrens, T.M.: Agent-oriented control in real-time computer games. In: Proceedings of ProMAS 2009 (2009)

3. Bordini, R., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.): Programming Multi Agent Systems: Languages, Platforms and Applications. Multiagent Systems, Artificial Societies and Simulated Organizations, vol. 15. Springer, Berlin (2005)
4. Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.): Multi-Agent Tools: Languages, Platforms and Applications. Springer, Berlin (2009)
5. Buro, M.: ORTS: A hack-free RTS game environment. In: Schaeffer, J., Müller, M., Björnsson, Y. (eds.) CG 2002. LNCS, vol. 2883, pp. 280–291. Springer, Heidelberg (2003)
6. Dastani, M.: 2APL: a practical agent programming language. Autonomous Agents and Multi-Agent Systems 16(3), 214–248 (2008)
7. Dastani, M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.): ProMAS 2007. LNCS (LNAI), vol. 4908. Springer, Heidelberg (2008)
8. Gonzalez, R.C., Woods, R.E.: Digital Image Processing, 3rd edn. Prentice-Hall, Inc., Upper Saddle River (2006)
9. Hagelbäck, J., Johansson, S.J.: Dealing with fog of war in a real time strategy game environment. In: Proceedings of 2008 IEEE Symposium on Computational Intelligence and Games, CIG (2008)
10. Hagelbäck, J., Johansson, S.J.: Using multi-agent potential fields in real-time strategy games. In: AAMAS (2), pp. 631–638 (2008)
11. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots, vol. 2, pp. 500–505 (1985)
12. Koren, Y. (Senior Member), Borenstein, J.: Potential field methods and their inherent limitations for mobile robot navigation. In: Proc. IEEE Int. Conf. Robotics and Automation, pp. 1398–1404 (1991)
13. Krogh, B.: A generalized potential field approach to obstacle avoidance control (1984)
14. Mamei, M., Zambonelli, F.: Field-based motion coordination in quake 3 arena. In: AAMAS, pp. 1532–1533. IEEE Computer Society, Los Alamitos (2004)
15. Massari, M., Giardini, G., Bernelli-Zazzera, F.: Autonomous navigation system for planetary exploration rover based on artificial potential fields. In: Proceedings of Dynamics and Control of Systems and Structures in Space (DCSSS) 6th Conference (2005)
16. Russell, S.J., Norvig: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Englewood Cliffs (2003)
17. Shirley, P., Ashikhmin, M., Gleicher, M., Marschner, S., Reinhard, E., Sung, K., Thompson, W., Willemsen, P.: Fundamentals of Computer Graphics, 2nd edn. A. K. Peters, Ltd., Natick (2005)
18. Weyns, D., Boucké, N., Holvoet, T.: A field-based versus a protocol-based approach for adaptive task assignment. Autonomous Agents and Multi-Agent Systems 17(2), 288–319 (2008)

Verification of Epistemic Properties in Probabilistic Multi-Agent Systems

Carla Delgado[1] and Mario Benevides[2,⋆]

[1] Fakultät für Informatik, Technische Universität Dortmund, Germany
carla.delgado@tu-dortmund.de
[2] Mathematics Institut, Universidade Federal do Rio de Janeiro, Brazil
mario@cos.ufrj.br

Abstract. Over the past decade Multi-Agent Systems (MAS) have emerged as a successful approach to develop distributed applications. In recent years proposals have been made to extend MAS models with probabilistic behavior. Languages to reason about such systems were presented in order to deal with uncertainty that can be encountered in practical application domains. While in recent works model checking techniques have been successfully applied for verifying knowledge in classical MAS, no methods for verifying knowledge in probabilistic MAS yet exist. This paper proposes such a model checking approach for probabilistic MAS. The approach comprises a compositional modeling process, a modal logic with operators for the specification of epistemic and temporal properties, the corresponding model checking procedure, and an outline of how these techniques can be implemented into existing model checking tools. The advantages of the chosen design include the possibility to analyze the MAS both from the global perspective as well as from the perspective of the agents, and the polynomial complexity of the model checking algorithm.

1 Introduction

Multi-agent systems (MAS) have gained importance as many distributed applications were developed on the last decade. MAS applications permeate a wide area such as decision support systems, business process management and electronic commerce [1,2,3]. According to [4] probabilistic MAS consist of "a collection of agents interacting in the presence of some source of randomness (such as a fair coin)". Examples of probabilistic MAS are peers in cryptographic and other randomized or probabilistic protocols that aim to provide guarantees about the probability of certain events.

As intelligent behavior is a central point in MAS, studies have been conducted to improve the understanding of the relations among autonomous behavior, uncertainty and interaction [5]. Significant work was done in order to obtain suitable logics to reason about epistemic aspects of MAS, often using a combination of the epistemic modal logic $S5_n$ and temporal logics [6,7,8]. The relations between knowledge and probability received special attention in [9,10], and recent approaches to support the specification and analysis of probabilistic MAS were presented in [11,12,13].

⋆ This work was partially funded by the agencies CAPES, CNPq and FAPERJ.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 16–28, 2009.

The behavior of a MAS is in general complex to predict, so formal means to analyze and verify such systems are required. Model checking is a successful approach to automatize the verification of computer systems and a recent growth of the application of these techniques to MAS has been observed. In particular, model checking of epistemic aspects of MAS received considerable attention during the last years [14,15,16,17,18,19]. However such techniques have not been applied yet to probabilistic MAS.

In this paper a first approach to verify probabilistic MAS is presented. As an example, consider a simple application with three agents: two image scanners (Agent A and Agent B) and a file transfer agent that collects the scanned images from the two scanners in order to send them to a processor (Agent C). The scanners have identical probabilistic behavior: they can either scan, calibrate or synchronize with agent C (respectively actions aRead, bRead, aCal, bCal, syncA and syncB). The models for the agents are used to build a global model for the system. Figure 1 shows the model for each agent and the global model obtained for the MAS. For the sake of simplicity the transmission actions of Agent C are not shown. The calibration and scan actions are probabilistic, so a probability value is associated to them. For instance, at state 0 of agent A's model aCal will be executed with probability 0.1 and aRead with probability 0.9. At the model for the complete MAS, a non-deterministic choice at state 000 will decide among A and B the next agent to perform an action.

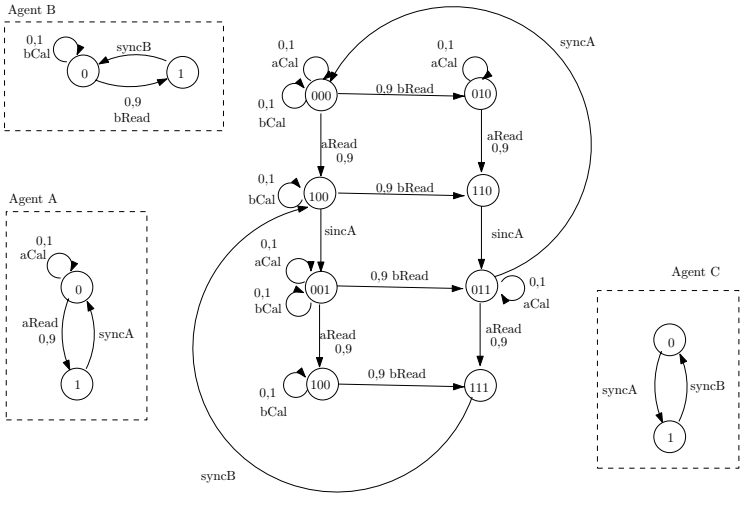


Fig. 1. A model for the MAS G_{ABC} is formed by composing the models of agents A, B and C

Assuming that the file transfer agent tries to optimize its throughput, it should reason about the probability that at least one scanner is ready to synchronize. The scanners might be interested in reasoning about how to behave if the file transfer agent is available for synchronization with probability lower than 0.4, so they can manage their activity schedule in order to avoid long waiting times. The specification and verification of properties that relate to the knowledge of agents in a probabilistic MAS are

the subject of this paper. We propose a modeling strategy, a specification language and the corresponding model checking algorithms, and describe how our approach can be implemented in the probabilistic model checker PRISM [20].

In recent years, several approaches to model and verify knowledge in MAS have been proposed: [21] discussed the problem in the context of systems with perfect recall, [19] provided a reduction of the model checking problem for the logic CKL_n (that combines LTL with epistemic logic) to LTL model checking, and more recently [22] presented a direct implementation for symbolic model checking of CKL_n. In [16] model checking approaches to the logic CKK_n (that combine CTL and epistemic logic) are provided and algorithms for model checking epistemic operators based on SMV are presented. Finally in [14] a technique for verifying an epistemic logic of branching time (CTLK) based on the NuSMV model checker [23] is presented.

Our work brings together previous results from the field of probabilistic model checking (e.g. [24,25,26,27]) and techniques for modeling and verifying knowledge in asynchronous MAS. We propose an epistemic extension of the probabilistic CTL temporal logic, called K-PCTL. K-PCTL allows for the expression of epistemic properties, temporal properties and likelihood of events. Our approach differs from the one in [11] as no probabilistic distribution is assigned to each agent-state pair in our formalism. In KPCTL probabilistic uncertainty is expressed using formulas with nested epistemic and probabilistic operators. In order to verify such epistemic formulas a model checking algorithm is provided in our work, and a description of how to extend PRISM to model check K-PCTL formulas over probabilistic MAS models is presented. Performance evaluation experiments were not in the scope of this work, but we prove that the verification process has polynomial complexity. Our main contributions are the extension of previous approaches to probabilistic MAS and a road-map for the implementation of the herein developed techniques in a powerful probabilistic model checker.

This paper is organized as follows. A modeling process for probabilistic MAS is presented in Section 2. In section 3 we present the logic K-PCTL and an interpretation of K-PCTL formulas. In Section 4 we discuss how to extend existing PCTL model checking processes to K-PCTL. Our final remarks and ideas to extend the results to continuous-time probabilistic MAS are stated in Section 5.

2 Modeling Probabilistic MAS

We are interested in modeling asynchronous probabilistic MAS. Such systems are composed of multiple agents that have probabilistic behavior and independent execution times. The general idea is to model the behavior of individual agents, and then compose these models to generate a model for the complete MAS. This will be done in order to obtain a global model for probabilistic MAS in the resemblance of the *interpreted systems* [7] formalism, but with a more precise characterization of the agents' behavior. Synchronization actions will be used to support interaction among agents.

In order to model agents that have probabilistic behavior, we will adopt a model that allows for the specification of probabilistic state changes. Markov Chains are a simple and adequate formalism for describing phenomena that change randomly as time progresses [26]. The simplest Markov Chain model is the Discrete Time Markov

Chain (DTMC). DTMCs are commonly used as models for probabilistic systems, as they provide a convenient representation of probabilistic behavior over time.

A DTMC is a collection of random variables $\{X(k)|k = 0, 1, 2, \ldots\}$ where observations are made at discrete time steps. The values taken by $X(k)$ are called *states*. The state space is discrete and for the scope of this work finite. A DTMC has the Markov Property, so that the probability that the system is at state s at time k is dependent only on the state of the system at time $k - 1$. A homogenous DTMC is a DTMC where the probability of a transition is independent of the time in which the transition occurs. Such DTMCs can be represented by a matrix that gives for each pair (s, t) of states the probability that a transition from s to t occurs.

In our approach each individual agent will be modeled by a homogeneous DTMC. For verification purposes, we enrich the model with the atomic propositions that are valid in each state. Throughout this section we use AP to denote a given a set of atomic propositions. Synchronization mechanisms are included in our model by means of special *synchronization actions*. As the system is asynchronous, one agent might have to wait for the other agents with which it has to synchronize until all are ready to perform the action together. For simplicity, we assume that agents perform synchronization actions always with probability 1. We define a DTMC with synchronization actions – *DTMC-SA* – as a tuple $\mathcal{D} = (S, s_0, Act, L, \mathbf{A}, \mathbf{P})$ where:

- S is a finite set of states;
- $s_0 \in S$ is the initial state;
- Act is a finite set of actions $\{a_1, \ldots a_n\}$;
- $L : S \to 2^{AP}$ is a labeling function describing the atomic propositions that are true in a state;
- $\mathbf{A} : S \times Act \times S \to \{0, 1\}$ describes the state changes caused by a synchronization action;
- $\mathbf{P} : S \times S \to [0, 1]$ is a matrix with rational values such that: $\sum_{s' \in S}(\mathbf{P}(s, s') + \mathbf{A}(s, a, s')) = 1$ holds for all $s \in S$ and $a \in Act$ (if a synchronization action is possible at a state, no other action is allowed).[1]

We now consider the construction of a model for the probabilistic MAS based on the models of the individual probabilistic agents. The probabilistic MAS model will be a parallel composition of the models representing its agents.

There are two main approaches for the composition of DTMCs: fully probabilistic and alternating [26]. Fully probabilistic approaches define a probability distribution that encompasses actions from the composed DTMCs. Alternating approaches interleave the actions from the composed DTMCs, which means that they do not require a "normalization" of distributions from the different DTMCs being composed. Interleaving actions from several DTMCs will introduce non-deterministic choices in the global model. The non-determinism stands for the freedom of choice regarding which agent from the MAS will be the next one to perform an action. This suits our model well as we are considering MAS whose agents are expected to operate independently and might have different execution times.

[1] Note that $\mathbf{P}$ is not a probability matrix as in the original DTMC model, though one could be generated adding to it the values of $\mathbf{A}$.

It is known that the alternating composition of a set of DTMC models results in a probabilistic and non-deterministic model called Markov Decision Process (MDP) (a detailed description of both formalisms appears in [25] and [26]). A MDP is a similar model to a DTMC, which instead of a probability matrix has a probabilistic transition function $R : S \rightarrow 2^{Act \times \text{Dist}(S)}$, where $\text{Dist}(S)$ is the set of all probability distributions over the set S.[2]

We can formalize the composition of DTMC-SAs representing probabilistic agents in order to obtain an MDP that represents the behavior of the overall MAS. In the next definition, we will use the fact that the function R can be rewritten as a relation $R \subseteq S \times Act \times \text{Dist}(S)$. For an overview of the composition process, see Figure 1.

Definition 1. *Let $\mathcal{P} = (S_\mathcal{P}, p_0, Act_\mathcal{P}, L_\mathcal{P}, \mathbf{A}_\mathcal{P}, \mathbf{P}_\mathcal{P})$ and $\mathcal{Q} = (S_\mathcal{Q}, q_0, Act_\mathcal{Q}, L_\mathcal{Q}, \mathbf{A}_\mathcal{Q}, \mathbf{P}_\mathcal{Q})$ be two DTMC-SAs. The parallel composition of $\mathcal{P}$ and $\mathcal{Q}$ synchronizing at actions $a_1, \ldots, a_n \in (Act_\mathcal{P} \cap Act_\mathcal{Q})$ is a MDP (S, s_0, Act, L, R) where:*

- $S = \{(pq) | p \in S_\mathcal{P}$ *and* $q \in S_\mathcal{Q}\}$*;*
- $s_0 = (p_0 q_0)$*;*
- $Act = \{a_\mathcal{P}, a_\mathcal{Q}\} \cup Act_\mathcal{P} \cup Act_\mathcal{Q}$*, where $a_\mathcal{P}$ and $a_\mathcal{Q}$ are two new action labels;*
- L *is the smallest relation satisfying that for every state* $(pq) \in S$*,* $L((pq)) = L_\mathcal{P}(p) \cup L_\mathcal{Q}(q)$*;*
- $R \subseteq S \times Act \times Dist(S)$ *is the relation defined by:*
  - *For each $p \in S_\mathcal{P}$ and $q \in S_\mathcal{Q}$:*
    if $\sum_{p' \in S_\mathcal{P}} \mathbf{P}_\mathcal{P}(p, p') = 1$ then $((pq), a_\mathcal{P}, \mu) \in R$, where μ is the distribution: $\mu(p'q') = 1$ if $q' = q$, and $\mu(p'q') = 0$ otherwise.
    if $\sum_{q' \in S_\mathcal{Q}} \mathbf{P}_\mathcal{Q}(q, q') = 1$ then $((pq), a_\mathcal{Q}, \mu) \in R$, where μ is the distribution: $\mu(p'q') = 1$ if $p' = p$, and $\mu(p'q') = 0$ otherwise.
  - *For each $a \in (Act_\mathcal{P} \cap Act_\mathcal{Q})$:*
    $((pq), a, \mu) \in R$, where μ is the distribution:

$$\mu(p'q') = \begin{cases} 1 & \text{if } (p, a, p') \in \mathbf{A}_\mathcal{P} \text{ and } (q, a, q') \in \mathbf{A}_\mathcal{Q}, \\ 0 & \text{otherwise} \end{cases}$$

  - *For each action $a \notin \{a_\mathcal{P}, a_\mathcal{Q}\} \cup (Act_\mathcal{P} \cap Act_\mathcal{Q})$:*
    $((pq), a, \mu) \in R$, where μ is the distribution

$$\mu(p'q') = \begin{cases} 1 & \text{if } (p, a, p') \in \mathbf{A}_\mathcal{P} \text{ and } q = q', \\ 1 & \text{if } (q, a, q') \in \mathbf{A}_\mathcal{Q} \text{ and } p = p', \\ 0 & \text{otherwise} \end{cases}$$

We will use the term "global model" to refer to a model obtained by a parallel composition of DTMC-SAs. The states of the global model are then ordered pairs representing the states of the individual agents' models . Considering that several agent models may be composed into one global model, we denote by $s[i]$ the i-th component of state s that corresponds to the local state of agent i in the global state s. New action labels $Act_\mathcal{P} \cup Act_\mathcal{Q}$ are introduced to denote the agent that was originally able to perform a

[2] $\text{Dist}(S)$ is the set of functions $\mu : S \rightarrow [0, 1]$ such that $\sum_{s \in S} \mu(s) = 1$.

probabilistic transition. The labeling function for a global sate (pq) is constructed in a way that the valid propositions are the ones that were valid at the states p and q from the respective agents' models. The probabilistic transitions are labeled with actions that denote the agent that is able to perform them; the synchronization transitions keep their previous labels. Note that at the time a composition is made not all synchronization transitions need to be synchronized, i.e. they can be left "free" as an open door to synchronization by a future composition.

The composition operation defined above can be extended for the composition of a collection of DTMC-SAs. Because of space limitations we will not present a formalization of this extension, but a short description of the composition process and of the resulting model. Let $Ag = \{1, \ldots, n\}$ be a set of DTMC-SAs. The states of the model generated by a parallel composition of the DTMC-SAs in Ag are tuples $(s_1, \ldots, s_n)$ where $s_i \in S_i$, being S_i the set of states of agent $i \in Ag$. The probabilistic transitions of the global model are constructed by interleaving the probabilistic transitions from all agents $i \in Ag$ with special attention to the synchronization actions. Such specific actions can be used to synchronize agents as specified in the composition operation, and are performed in parallel by the agents that are synchronizing.

In spite that the model for each agent is fully probabilistic, the composed model has completely non-deterministic choices, to which no probability distribution is assigned. The probabilistic distributions that govern the behavior of individual agents do not influence one another. The non-determinism is generated exactly when two different distributions "compete" in one state, which happens when two different agents perform probabilistic actions and independently change their states.

The goal of a MAS is usually defined in terms of global behavior, whereas its execution is the result of the concurrent execution of all agents. The actions of an agent are chosen each step by the agent itself, taking into account its available information. We may say that the information an agent has available and is able to reason about is its *knowledge*. Usually an agent does not know everything about a MAS, as some information may not be known at all or may be unavailable to the agent at a certain time, such as the outcome of events under the control of other agents. This notions will be represented by means of an equivalence relation $\sim_i$, defined for each agent i over the set of states of the global model. The intuition behind the $\sim_i$ relations is to connect states of the global model where an agent i has the same the local state (e.g. states 000 and 010 in Figure 1 for the agents A and C). This means that the knowledge agent i has in both states is the same, and the agent is not able to distinguish between them (states related by $\sim_i$ are called "indistinguishable"). Agent i is said to know a fact if and only if the fact holds in all states i considers possible, that is, all states that are related to the current state by $\sim_i$ [7].

Let $\mathfrak{A} = (S, s_0, Act, L, R)$ be the parallel composition of a collection of DTMC-SAs $\{\mathfrak{A}_i\}$ for $i \in [1, \ldots, n]$. The accessibility relation $\sim_i \in S \times S$ for each agent i is the smallest equivalence relation containing all pairs (s, s'), where s and $s' \in S$ such that the local state of agent i is the same both at s and s': $s'[i] = s[i]$. We call the resulting model MDP$_\sim$. A MDP$_\sim$ is then a global model for a MAS that encapsulates behavior information (by means of state transitions) and epistemic information (by means of the relations $\sim_i$), and can be fully constructed from composing probabilistic agent models.

3 The Logic K-PCTL

In this section we consider how to formally specify epistemic properties of a probabilistic MAS. Different logics to reason about knowledge and probability were already proposed. [11] proposes a logic that combines probability and epistemic operators. This logic is extended in [13] in order to deal with information change. Both logics require that a probability function is assigned to each agent of the system for each state. These distributions are used to denote, for a given state, the probability that an agent attributes to the other states of the system. Such distributions are not easy to obtain directly from the model, particulary for asynchronous systems. The biggest obstacle is that no agent knows how quick the other agents run, and the chances that a specific state will be the next one during an execution depends on the state changes of all agents.

Here we take a different approach and use PCTL as a base language. The main reasons for this choice are that PCTL is commonly used to reason about time in probabilistic models of discrete time as DTMCs and MDPs [28,25,29] and that the corresponding model checking algorithms were already investigated. PCTL extends Real Time CTL – RCTL [30], which is itself an extension of the temporal logic CTL [31,32]. As pointed out in [33], RCTL can be used to state *hard deadlines*, while PCTL allows *soft deadlines*; RCTL allows to express properties such as *"A property holds within a certain amount of (discrete) time"* and PCTL allows to state properties such as *"A property holds within a certain amount of (discrete) time with a probability of at least 99%"*.

We propose a combination of PCTL with the basic modal epistemic logic $S5_n$[7], in order to allow the expression of properties about probabilistic behavior and the (not probabilistic) knowledge of the agents. We call this language K-PCTL. It includes operators K_i for each agent i of the MAS and epistemic operators for group knowledge and common knowledge (E_G and C_G). The intuitive meaning of formulas $K_i\phi$ is "Agent i knows ϕ". $E_G\phi$ means that all agents of the group G knows ϕ, and $C_G\phi$ represents that ϕ as a convention among the agents of the group G. K-PCTL makes it possible to express properties about knowledge, time and events in MAS, as for example: "Agent A knows that if he sends a message to agent B, the probability that B will eventually know the content of this message is 0.998". This can be expressed by the K-PCTL formula $K_A(\ \texttt{msgsent} \rightarrow \mathcal{P}_{\geq 0.998}(\exists \mathsf{F} K_B\ \texttt{msg}\ ))$, where the proposition `msgsent` stands for "agent A sent the message `msg` to agent B" and the proposition `msg` stands for the content of the message.

The syntax is given in terms of state formulas ϕ and path formulas ψ that are evaluated respectively over states and paths[3]. Let A be the set of agents.

$$\phi ::= true \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{P}_{\mathrm{relp}}[\psi] \mid K_i\phi \mid C_G\phi \mid E_G\phi$$
$$\psi ::= \phi \mid \phi \mathcal{U}^{\leq k} \phi \mid \phi \mathcal{U} \phi,$$

where a is an atomic proposition, $\mathrm{rel} \in \{\leq, <, \geq, >\}$, $p \in [0,1]$, $c \in \mathbb{R}^{\geq 0}$, $i \in A$ and $k \in \mathbb{N}$.

Properties over the model will always be expressed using state formulas, which may represent boolean combinations of atomic propositions, probabilistic properties ($\mathcal{P}_{\mathrm{relp}} f$) that require a certain measure of existing paths starting in the current state and

[3] The cost operator $\mathcal{E}_{\mathrm{relc}}[\phi]$ was omitted for the sake of simplicity, but it can be directly introduced in K-PCTL as long as a cost structure is included in the model to support its semantics.

satisfying the path formula f, or epistemic formulas like $K_i\phi$. Path formulas involve the operator "until" $\mathcal{U}$ or bounded $\mathcal{U}^{\leq k}$ and the "next step" path (represented by the path formula ϕ). Other boolean operators can be derived from $\wedge$ and $\neg$ in the usual way. Intuitively the path formula $\phi_1 \mathcal{U}^{\leq k} \phi_2$ means that ϕ_1 holds from now on until within at most k time units ϕ_2 becomes true. $\phi_1 \mathcal{U} \phi_2$ means that ϕ_1 holds from now on until ϕ_2 becomes true. Adversaries are used to deal with non-determinism in order allow for the computation of probabilities over paths. An adversary is basically a policy that determines a non-deterministic choice based on the previous path (so on the history of choices made so far) [25].

We present an interpretation for K-PCTL formulas based on the interpretation for PCTL over MDP$\sim$ models. The semantics of the epistemic modalities is based on the accessibility relations $\sim_i$.

$$
\begin{array}{ll}
s \models true & \text{for all } s \in S \\
s \models a & \text{if and only if } a \in \mathcal{L}(s) \\
s \models \neg\phi & \text{if and only if } s \not\models \phi \\
s \models \phi_1 \wedge \phi_2 & \text{if and only if } s \models \phi_1 \text{ and } s \models \phi_2 \\
s \models K_i\phi & \text{if and only if for all } t \in S \mid t \sim_i s \Rightarrow t \models \phi \\
s \models E_G\phi & \text{if and only if } s \models K_i\phi \text{ for all } i \in G \\
s \models C_G\phi & \text{if and only if } s \models E_G^k\phi \text{ for all } k \geq 1 \\
s \models \mathcal{P}_{\mathrm{rel}p}[\psi] & \text{if and only if } p_s^A(\psi)\mathrm{rel}p \text{ , for all } A \in Adv_{\mathcal{M}}
\end{array}
$$

where for all adversaries $A \in Adv_{\mathcal{M}}$:

$$p_s^A(\psi) \stackrel{\text{def}}{=} Prob_s^A(\{w \in Path_s^A | w \models \psi\})$$

and for each path $w \in Path$:

$$
\begin{array}{ll}
w \models \phi & \text{if and only if } w(1) \models \phi, \\
w \models \phi_1 \mathcal{U}^{\leq k} \phi_2 & \text{if and only if } \exists i \leq k \cdot (w(i) \models \phi_2 \wedge w(j) \models \phi_1, \forall j < i), \\
w \models \phi_1 \mathcal{U} \phi_2 & \text{if and only if } \exists k \geq 0 \cdot w \models \phi_1 \mathcal{U}^{\leq k} \phi_2.
\end{array}
$$

The set of paths $\{w \in Path_s^A | w \models \psi\}$ is measurable for any path formula ψ, state $s \in S$ and adversary A [34]. The operators false, $\vee$ and $\rightarrow$ can be defined using true, $\neg$ and $\wedge$. To illustrate the expressiveness of K-PCTL let us reconsider the example MAS from Figure 1. Consider that in the local state 0 of agent A and B propositions $calA$ and $calB$ are (respectively) valid, indicating that the agent is calibrating. The formula K-PCTL $K_C P_{\leq 0,1} calA \wedge calB$ expresses that Agent C knows that the probability that both agents A and B are calibrating is less then 0.1.

4 Verification

Now we discuss how to verify knowledge formulas like $K_i\phi$ from K-PCTL against global models for MAS. As our intention is to benefit from the existence of a powerful probabilistic model checker for PCTL, let us first consider how to map our modeling approach to PRISM. PRISM is a probabilistic model checker that supports three types of probabilistic models: DTMCs, MDPs and also continuous-time Markov chains

(CTMCs), plus extensions of these models with costs and rewards, and it has already been used for a large and diverse set of case studies. For a more detailed description of PRISM and its functionalities refer to [20].

The herein presented modeling techniques are completely compatible with the latest PRISM implementation: the models for each agent can be described independently and the available composition mechanism provides a global model compatible with the one given by the parallel composition operation from Definition 1 as long as the agent models are declared as MDPs (and not as DTMCs as one could expect[4]). This is a mere technicality and is not a problem as DTMCs are a subclass of MDP models. The use of synchronization actions is also supported by PRISM. Although PRISM does not enforce that these actions have probability 1, we can consider without loss of generality that this restriction will be enforced during the modeling process. As it is common in PRISM, we can use a variable to represent the state of an agent. For the global model, the state will be represented by the set of the state variables of all agents. This representation suits our model as it keeps track of the local states and can then be used to find out the states related by $\sim_i$.

The general model checking process adopted by PRISM takes as an input a formula ϕ from PCTL and an MDP model M and checks if ϕ holds in each state of M. The list of states where ϕ holds can then be obtained. Roughly speaking, this is done by constructing a parse tree for formula ϕ, where each node is labeled with a subformula from ϕ. ϕ is the label of the root of the tree and atomic propositions are the leaves. The states satisfying each subformula are computed working upwards towards the root of the tree.

To extend the model checking process to K-PCTL formulas it is enough to provide an algorithm for handling the operator K_i as the algorithms for the epistemic operators E_G and C_G are based on the verification of K_i formulas (such algorithms will not be presented here because of space restrictions) and all remaining operators can be treated by the existing algorithms of PCTL model checking. A straightforward implementation of an algorithm for checking $K_i\phi$ would proceed as follows: "for each state $s \in S$, for each state s' related to s by $\sim_i$, check if it is the case that ϕ holds. If yes, then $K_i\phi$ holds for s". This basic approach can be improved by taking advantage of the following facts:

- By the time the formula $K_i\phi$ has to be computed, the set $T(\phi)$ of states of M where ϕ holds has already been computed.
- Given a state s and an agent i we can obtain the set $T(s \sim_i)$ of states s' such that $s \sim_i s'$ by checking for which states of the model $s'[i] = s[i]$ holds. Once s is fixed this is a simple task as it means to check for the states where a variable has a specific value.
- As $\sim_i$ is by definition an equivalence relation and considering the semantics given for $K_i\phi$, it is the case that for all states where $K_i\phi$ holds, ϕ must also hold, and $K_i\phi$ holds at a state s if and only if it holds for all states s' such that $s \sim_i s'$ [16].

Let $T(K_i\phi)$ denotes the set of states where $K_i\phi$ holds, and $T(\phi)$ denotes a set of states where ϕ holds. Now consider that $T(\phi)$ is already known by the time K_i has to be

[4] The composition operation defined for DTMCs at PRISM returns a DTMC and not an MDP.

handled (as part of the normal model checking process executed by PRISM); consider also that PRISM can generate the set $T(s \sim_i)$. Then a convenient approach to obtain $T(K_i\phi)$ is given by the following algorithm `Check-K`:

- For each state s from $T(\phi)$:
  - Obtain the set $T(s \sim_i)$ of states s' such that $s \sim_i s'$ (as described above).
  - Check whether $T(s \sim_i) \subseteq T(\phi)$. If yes, then include $T(s \sim_i)$ in $T(K_i\phi)$. In either case remove all states that appear in $T(s \sim_i)$ from $T(\phi)$.
- return $T(K_i\phi)$.

The algorithm has to traverse the states s of $T(\phi)$ checking whether $T(s \sim_i) \subseteq T(\phi)$. For each state s that is checked, all states from $T(s \sim_i)$ are removed from $T(\phi)$. When $T(\phi)$ is empty the set $T(K_i\phi)$ is returned and the algorithm ends. A proof of correctness of this algorithm follows from the one presented in [16].

Proposition 1. *The algorithm* `Check-K` *always terminates.*

Proof. The number of states in $T(\phi)$ is finite and limited by S. Each state of $T(\phi)$ is visited at most once. Constructing $T(s \sim_i)$ can be done in finite number of steps, as well as checking if $T(s \sim_i) \subseteq T(\phi)$.

Proposition 2. *The time complexity to obtain $T(K_i\phi)$ given $T(\phi)$ according to the algorithm* `Check-K` *is $O(|S|^2)$, where $|S|$ is the number of states of the global model.*

Proof. The proof presented in [16] for the non-probabilistic case can be applied here with no loss of generality. The complexity to obtain $T(s \sim_i)$ is $O(|S|)$. To traverse the set $T(\phi)$, we must check in every step whether $T(s \sim_i) \subseteq T(\phi)$ and then delete the states of $T(s \sim_i)$ from $T(\phi)$. Each of these operations takes at most $|T(s \sim_i)| * |S|$ steps. Assume that it takes k steps until $T(\phi)$ becomes empty; it follows that $T = \cup_{j=1}^{k} T(s_j \sim_i)$ where s_j is the state being analyzed in step j. Also, $T(s_j \sim_i) \cap T(s_l \sim_i) = \emptyset$ for any j, l from 1 to k. To finish traversing $T(\phi)$ it takes $\sum_{j=1}^{k}(|S| + 2 * |T(s_j \sim_i)| * |S|)$ steps, what is clearly less then $k * |S| + 2 * |S|^2 < 3 * |S|^2$ and so polynomial in the size of the global model.

5 Conclusion

In this paper we have considered the problem of model checking knowledge and time in probabilistic MAS. Based on previous results for model checking of knowledge logics against non-probabilistic MAS, we described a concise approach for modeling a probabilistic MAS, a logic to specify knowledge properties, and the corresponding process for model checking epistemic formulas. Our approach is at the same time theoretically sound and suitable for implementation. A road-map was provided describing how PRISM can be used to verify knowledge specifications against MDP models representing a probabilistic MAS, what is as well a description of how PRISM could be extended to incorporate this functionality.

In our search for meaningful ways to express knowledge properties of probabilistic MAS we found out that the assumption of time-independence among the agents is very

strong. If an agent has no extra information about the behavior of the others, it has no reasons to consider that some states are most likely to be the current one than others. The question "which is the current state at the moment" is linked to the history of actions performed by all agents. As actions are interleaved, this type of uncertainty is non-deterministic from the point of view of one agent. It is though sill possible for an agent to reason about the probability that some property will eventually hold, as K-PCTL provides mechanisms to reason about the probabilities a path will be reached by means of quantifying over all possible non-deterministic choices (adversaries) with the probabilistic operator $\mathcal{P}$.

Regarding the knowledge properties, our approach was to treat them as indistinguishable states. If adversaries should be fixed determining a policy to solve the non-deterministic choices we could as well use this new information to refine the agents' knowledge, and build probabilistic accessibility relations. If we go one step further and consider continuous-time probabilistic models like Continuous Time Markov Chains (CTMC), we face a scenario where agents have a global notion of the time passage (though their execution rates might freely differ). This fact can be used in order to build more refined accessibility relations. This is the focus of our work at the moment and our preliminary results are presented in [35].

References

1. Arus, C., Celda, B., Dasmahaptra, S., Dupplaw, D., Gonzalez-Velez, H., Huffel, S.V., Lewis, P., Ariet, M.L.i., Mier, M., Peet, A., Robles, M.: On the design of a web-based decision support system for brain tumour diagnosis using distributed agents. In: Proc. IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology, pp. 208–211 (2006)
2. Duo, W., Yi, L., Wenhui, L., Qi, J., Rongqing, Y.: Intelligent multi-agent based information system of business process management. In: Pacific-Asia Workshop on Computational Intelligence and Industrial Application, pp. 469–473 (2008)
3. Gleizes, M.P., Link-Pezet, J., Glize, P.: An adaptive multi-agent tool for electronic commerce. In: IEEE Int. Workshops on Enabling Technologies, vol. 1, pp. 59–66 (2000)
4. Halpern, J.Y., Tuttle, M.R.: Knowledge, probability, and adversaries. J. ACM 40(4), 917–960 (1993)
5. Wooldridge, M.J.: Introduction to Multiagent Systems. John Wiley & Sons, Inc., Chichester (2001)
6. Lehmann, D., Shelah, S.: Reasoning with time and chance. Information and Control 53, 165–198 (1982)
7. Fagin, R., Halpern, J.Y., Moses, Y.: Reasoning about knowledge. MIT Press, Cambridge (1995)
8. der Hoek, W.V., Wooldridge, M.: Cooperation, knowledge and time: Alternating-time temporal epistemic logic and its applications. Studia Logica 75(1), 125–157 (2003)
9. Halpern, J.Y.: Reasoning about Uncertainty. MIT Press, Cambridge (2003)
10. Grünwald, P., Halpern, J.Y.: A game-theoretic analysis of updating sets of probabilities. In: Proc. 24th Conf. in Uncertainty in Artificial Intelligence, pp. 240–247 (2008)
11. Fagin, R., Halpern, J.Y.: Reasoning about knowledge and probability. J. ACM 41(2), 340–367 (1994)
12. de Carvalho Ferreira, N., Fisher, M., van der Hoek, W.: Specifying and reasoning about uncertain agents. International Journal of Approximate Reasoning 49(1), 35–51 (2008)

13. Kooi, B.P.: Probabilistic dynamic epistemic logic. J. of Logic, Lang. and Inf. 12(4), 381–408 (2003)
14. Lomuscio, A., Pecheur, C., Raimondi, F.: Automatic verification of knowledge and time with NuSMV. In: Proc. 20th IJCAI, pp. 1384–1389 (2007)
15. Lomuscio, A., Raimondi, F.: Model checking knowledge, strategies, and games in multi-agent systems. In: Proc. 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 161–168. ACM Press, New York (2006)
16. Wu, L., Su, K., Chen, Q.: Model checking temporal logics of knowledge and its application in security verification. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 349–354. Springer, Heidelberg (2005)
17. Gammie, P., van der Meyden, R.: Mck: Model checking the logic of knowledge. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 479–483. Springer, Heidelberg (2004)
18. Kacprzak, M., Lomuscio, A., Penczek, W.: Verification of multiagent systems via unbounded model checking. In: Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 638–645. ACM Press, New York (2004)
19. van der Hoek, W., Wooldridge, M.: Model checking knowledge and time. In: Bošnački, D., Leue, S. (eds.) SPIN 2002. LNCS, vol. 2318, pp. 95–111. Springer, Heidelberg (2002)
20. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 441–444. Springer, Heidelberg (2006)
21. van der Meyden, R., Shilov, N.V.: Model checking knowledge and time in systems with perfect recall (extended abstract). In: Pandu Rangan, C., Raman, V., Sarukkai, S. (eds.) FST TCS 1999. LNCS, vol. 1738, pp. 432–445. Springer, Heidelberg (1999)
22. Su, K., Sattar, A., Luo, X.: Model checking temporal logics of knowledge via OBDDs. The Computer Journal 50(4), 403–420 (2007)
23. Cimatti, A., Clarke, E., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An opensource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, p. 359. Springer, Heidelberg (2002)
24. Dekhtyar, M.I., Dikovsky, A.J., Valiev, M.K.: Temporal verification of probabilistic multi-agent systems. In: Avron, A., Dershowitz, N., Rabinovich, A. (eds.) Pillars of Computer Science. LNCS, vol. 4800, pp. 256–265. Springer, Heidelberg (2008)
25. Rutten, J., Kwiatkowska, M.Z., Norman, G., Parker, D.: Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems. CRM Monograph Series, vol. 23. American Mathematical Society, Providence (2004)
26. Hermanns, H.: Interactive Markov Chains: The Quest for Quantified Quality. Springer, Heidelberg (2002)
27. Katoen, J.P., Kwiatkowska, M.Z., Norman, G., Parker, D.: Faster and symbolic CTMC model checking. In: de Luca, L., Gilmore, S. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, pp. 23–38. Springer, Heidelberg (2001)
28. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
29. Aziz, A., Singhal, V., Balarin, F.: It usually works: The temporal logic of stochastic systems. In: Wolper, P. (ed.) CAV 1995. LNCS, vol. 939, pp. 155–165. Springer, Heidelberg (1995)
30. Emerson, E.A., Mok, A.K., Sistla, A.P., Srinivasan, J.: Quantitative temporal reasoning. In: Clarke, E., Kurshan, R.P. (eds.) CAV 1990. LNCS, vol. 531, pp. 136–145. Springer, Heidelberg (1991)

31. Ben-Ari, M., Manna, Z., Pnueli, A.: The temporal logic of branching time. In: Proc. 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 164–176. ACM Press, New York (1981)
32. Peled, D.A., Clarke, E.M., Grumberg, O.: Model Checking. MIT Press, Cambridge (2000)
33. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004)
34. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite state programs. In: Proc. 26th Annual Symposium on Foundations of Computer Science, pp. 327–338. IEEE, Los Alamitos (1985)
35. Delgado, C.: Modelagem e verificação de propriedades epistêmicas em sistemas multi-agentes. PhD thesis, Universidade Federal do Rio de Janeiro - UFRJ (2007)

GOAL as a Planning Formalism

Koen V. Hindriks and Tijmen Roberti

Delft University of Technology, Delft, The Netherlands

Abstract. It has been observed that there are interesting relations between planning and agent programming. This is not surprising as agent programming was partially motivated by the lack of planners that are able to operate in dynamic, complex environments. Vice versa it has also been observed, however, that agent programming languages typically lack planning capabilities. We show in this paper that the agent programming language GOAL is not only a programming language but can actually be used as a planning formalism as well. This opens up many possibilities for various approaches to mix execution and planning in agent-oriented programming. Moreover, by using the recently introduced *temporal* GOAL we are able to include not only the stratified axioms and ADL that are part of PDDL but also plan constraints.

1 Introduction

It has been argued in e.g. [3,13,14] that the combination and integration of planners into agent programming languages has many benefits. By combining the strengths of planners with the flexibility of agent programs it may be possible to handle dynamic domains more effectively, and, moreover, it becomes possible to exploit the advances of automated planning in agent programming.

Existing work on formally relating programming languages and planning formalisms, as far as we know, has mainly focussed on the language Golog. For example, in [13] the relative expressiveness of ADL [12] and Golog is investigated and a maximal fragment of so-called *basic action theories* used in Golog is identified that is expressively equivalent to ADL. We are not aware, however, of any work that formally relates agent programming languages that are based on BDI concepts such as beliefs and goals to planning from first principles.

The main contribution of this paper is to formally show that the agent programming language GOAL [10] can be used as a planning formalism. GOAL agents are BDI agents that derive their choice of action from their beliefs and goals. More specifically, we show that a fragment of PDDL including *axioms*, *ADL* and *(temporal) plan constraints* [8] can be compiled into GOAL. This result is important because it provides a clear interface between agent programs and planners. Such an interface can be used to call planners from agent programs and, vice versa, import results from research on planning into agent programming languages. It also shows that GOAL agents can solve solvable PDDL problems for a PDDL fragment including conditional effects and plan constraints. Moreover, it clarifies the overlap and difference in concepts present in agent programming languages and planning formalisms.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 29–40, 2009.

A planning formalism uses an underlying knowledge representation (e.g. first-order logic in PDDL). We introduce the notion of a GOAL *framework* to separate the underlying knowledge representation used by GOAL agents from the features of the language GOAL itself. This allows us to clearly separate the features of the GOAL language that have been used to obtain our results from those provided by the knowledge representation language. We believe that this approach may also be helpful in clarifying how our results may be applied to other agent programming languages that have declarative beliefs and goals.

The paper is organized as follows. In Section 2 we introduce the syntax and semantics of the PDDL 3.0 Level 1 fragment (without preferences). Section 3 introduces the agent programming language GOAL. In Section 4 we show how PDDL problems can be compiled into GOAL. Section 5 concludes the paper.

2 PDDL Fragment: Axioms + ADL + Plan Constraints

The fragment of PDDL that we consider here is PDDL 3.0 Level 1, including axioms, ADL, and plan constraints, but excluding preferences. We will refer to this fragment as $PDDL_{Ax+CE+PC}$.

We assume that a first-order function-free language $\mathcal{L}_0$ is given that is built from a set of predicates $\mathcal{P}$, variables $\mathcal{V}$, and constants $\mathcal{C}$ in the usual way and includes equality $=$. Note that constants are allowed although $\mathcal{L}_0$ is otherwise assumed to be function-free. The set of predicates is divided into two disjunct sets of so-called *basic* predicates $\mathcal{B}$ and *derived* predicates $\mathcal{D}$. PDDL also allows typed variables but for reasons of space we do not discuss this feature. Formulae $\phi \in \mathcal{L}_0$ are also called *state formulae*, and we write $\phi[\boldsymbol{x}]$ to indicate that all free variables of ϕ occur in the vector $\boldsymbol{x}$. State formulae are used in PDDL to define the goal state $\mathcal{G}$ and as precondition of action operators. It is common to also call state formulae *goal descriptions*. We define PDDL axiom sets as in [15].

Definition 1. *(PDDL Axiom)*
A PDDL axiom *is a closed formula of the form* $\forall \boldsymbol{x}.\phi \rightarrow d(\boldsymbol{x})$*, where* $d \in \mathcal{D}$ *and* $\phi \in \mathcal{L}_0$ *(whose free variables are in the vector* $\boldsymbol{x}$*).*

PDDL axioms in this context are best thought of as definitions of derived predicates d (cf. the completion of axioms in the sense of [1]). A closed world semantics is used (see below) where $d(\boldsymbol{x})$ is false whenever it cannot be derived as true. In order to define PDDL axiom sets we use the notion of a *Negation Normal Form (NNF)*. A formula $\phi \in \mathcal{L}_0$ is in negation normal form iff negation occurs directly in front of atoms.

Definition 2. *(PDDL Axiom Set)*
A PDDL Axiom Set *is a set of PDDL axioms that is stratified. A PDDL axiom set* $\mathcal{A}$ *is stratified iff there exists a partition of the set of derived predicates* $\mathcal{D}$ *into (non-empty) subsets* $\{\mathcal{D}_i, 1 \leq i \leq n\}$ *such that for every* $d_i \in \mathcal{D}_i$ *and every axiom* $\forall \boldsymbol{x}.\phi \rightarrow d_i(\boldsymbol{x}) \in \mathcal{A}$ *we have that:*

1. *if* $d_j \in \mathcal{D}_j$ *occurs positively in an NNF of* ϕ*, then* $j \leq i$
2. *if* $d_j \in \mathcal{D}_j$ *occurs negated in an NNF of* ϕ*, then* $j < i$

The semantics of $\mathcal{L}_0$ is defined relative to a state and axiom set [15]. A PDDL *state* S is set of ground positive literals from $\mathcal{B}$. The closed world assumption applies, so any ground positive literal not in S is assumed to be false. Axioms, however, need to be treated separately and we first assume the consequences of axioms $\mathcal{A}$ are given by a set D of atoms of the form $d(\boldsymbol{x})$ with $d \in \mathcal{D}$.

Definition 3. *(Semantics of $\mathcal{L}_0$)*
Let $\mathcal{C}$ be the constants of $\mathcal{L}_0$, S be a PDDL state, D a set of atoms $d(\boldsymbol{x})$ with $d \in \mathcal{D}$, and $\mathcal{A}$ an axiom set. We only provide the most important clauses:

$$\begin{array}{ll} \langle S, D\rangle \models p(\boldsymbol{t}) & \textit{iff } p(\boldsymbol{t}) \in S \cup D \\ \langle S, D\rangle \models \neg\phi & \textit{iff } \langle S, D\rangle \not\models \phi \\ \langle S, D\rangle \models \forall x.\phi[x] & \textit{iff } \langle S, D\rangle \models \phi[c] \textit{ for all } c \in \mathcal{C} \end{array}$$

Intuitively, we can derive $d(\boldsymbol{t})$ using axiom $a = \forall \boldsymbol{x}.\phi \rightarrow d(\boldsymbol{x})$ if we have $\langle S, D\rangle \models \phi[\boldsymbol{t}]$, and add $d(\boldsymbol{t})$ to D if not already present; we write $[\![a]\!](S, D) = \{d(\boldsymbol{t}) \mid \langle S, D\rangle \models \phi[\boldsymbol{t}], \boldsymbol{t} \text{ is ground}\}$ to denote these consequences. Then the set of consequences of an axiom set $\mathcal{A}$ can be computed as follows, assuming that we have a stratification $\{A_i, 1 \leq i \leq n\}$ of $\mathcal{A}$ (cf. [15]): define $[\![\mathcal{A}]\!]_0(S) = \emptyset$, and, for $1 \leq i \leq n$, define

$$[\![\mathcal{A}]\!]_i(S) = \bigcap \left\{ D \mid \bigcup_{a \in A_i} [\![a]\!](S, D) \cup [\![A]\!]_{i-1}(S) \subseteq D \right\}$$

Finally, $S \models_{\mathcal{A}} \phi$ is defined as $\langle S, [\![\mathcal{A}]\!](S)\rangle \models \phi$.

Action operators in the ADL fragment of PDDL specify the preconditions and (conditional) effects of actions. Performing an action changes the state S.

Definition 4. *(Action Operators)*
A PDDL action operator α *is a triple $\langle \boldsymbol{x}, \pi_\alpha, \epsilon_\alpha\rangle$ where $\boldsymbol{x}$ are the action's parameters, $\pi_\alpha \in \mathcal{L}_0$ defines when the action can be (successfully) executed, and ϵ_α is set of conditions of the form $\phi \Rightarrow \delta$ with $\phi \in \mathcal{L}_0$ and δ a set of literals. All free variables in π_α and ϵ_α must also occur in $\boldsymbol{x}$.*

The effect of an action α on a state S can be derived by computing the positive effects Eff^+_{ADL} and negative effects Eff^-_{ADL}. Given that $pos(\delta)$ and $neg(\delta)$ return respectively the positive and negative literals in δ, these are defined by:

$$\mathit{Eff}^+_{ADL} = \{l \in pos(\delta) \mid S \models_{\mathcal{A}} \phi, \phi \Rightarrow \delta \in \epsilon_\alpha\}$$

$$\mathit{Eff}^-_{ADL} = \{l \in neg(\delta) \mid S \models_{\mathcal{A}} \phi, \phi \Rightarrow \delta \in \epsilon_\alpha\}$$

If the precondition of a ground operator α with effect ϵ_α holds in the current state, i.e. $S \models_{\mathcal{A}} \pi_\alpha$, then the successor state $\gamma(S, \alpha)$ is defined by:

$$\gamma(S, \alpha) = (S \setminus \mathit{Eff}^-_{ADL}) \cup \mathit{Eff}^+_{ADL}$$

A *plan* π, which is a sequence of action operators $\langle \alpha_0, \ldots, \alpha_{n-1}\rangle$, generates a sequence of states: $\langle S_0, \ldots, S_n\rangle = \langle S_0, \gamma(S_0, a_0), \gamma(S_1, a_1), \ldots, \gamma(S_{n-1}, a_{n-1})\rangle$. Such a sequence is also called a *state trajectory*.

The language $\mathcal{L}_0$, axioms, and action operators are combined into a *planning domain definition* Δ, which is formally defined by $\Delta = \langle \mathcal{L}_0, \mathcal{A}, \mathcal{O} \rangle$, where $\mathcal{L}_0$ is a first-order function-free language based on $\mathcal{B}$, $\mathcal{D}$, $\mathcal{V}$, and $\mathcal{C}$, $\mathcal{A}$ is a stratified axiom set, and $\mathcal{O}$ is a set of action operators.

The final part of the PDDL fragment that we consider here are plan constraints (we do not discuss preferences). Plan contraints are like goals but apply to the state trajectory of a plan instead of only to the final state. A limited number of temporal modalities to express constraints are available, and we omit modalities that require explicit reference to time such as the `within` modality.

Definition 5. *(Plan Constraint)*
A plan constraint Φ *is defined as follows:* $\Phi = X\phi \mid \Phi \wedge \Phi \mid \forall \boldsymbol{x}.\Phi[\boldsymbol{x}]$, *where* X *is one of the modalities* `at end`, `always`, `sometime`, `at-most-once`, `sometime-after`, `sometime-before` *and* $\phi \in \mathcal{L}_0$ *a state formula.*

In contrast with standard linear temporal logic, plan constraints are evaluated relative to a *finite* state trajectory $\langle S_0, \ldots, S_n \rangle$ generated by a plan, and, as before, an axiom set $\mathcal{A}$.

Definition 6. *(Semantics of Plan Constraints)*
The semantics of temporal constraints is defined by the following clauses:

$$
\begin{array}{lll}
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{at end}\ \phi & \textit{iff} & S_n \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{always}\ \phi & \textit{iff} & \forall i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime}\ \phi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{at-most-once}\ \phi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \neg \exists j, k : i < j < k \leq n : S_j \models_{\mathcal{A}} \neg\phi \,\&\, S_k \models_{\mathcal{A}} \phi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime-after}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \exists j : i < j \leq n : S_j \models_{\mathcal{A}} \psi \\
\langle S_0, \ldots, S_n \rangle \models_{\mathcal{A}} \texttt{sometime-before}\ \phi\ \psi & \textit{iff} & \exists i : 0 \leq i \leq n : S_i \models_{\mathcal{A}} \phi \Rightarrow \\
 & & \exists j : 0 \leq j < i : S_j \models_{\mathcal{A}} \psi
\end{array}
$$

We now have all the ingredients that are needed to define a *PDDL problem*. A PDDL problem extends a domain with more specific information regarding the initial state (which literals are true and false initially) and the goal the plan should achieve. Additionally, plan constraints may be provided that must also be satisfied by a plan.

Definition 7. *(PDDL Problem)*
A $PDDL_{Ax+CE+PC}$ planning problem Π *is a tuple* $\langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$, *where* Δ *is a domain definition,* $\mathcal{C}$ *is a set of constants,* $\mathcal{I}$ *is the initial state,* $\mathcal{G} \in \mathcal{L}_0$ *is a closed formula called the* goal description, *and* $\mathcal{PC}$ *is a set of plan constraints.*

A plan π is said to be a *solution* for a planning problem Π iff the plan can be executed, the associated plan constraints are satisfied by the state trajectory of the plan and the goal description $\mathcal{G}$ of the problem is satisfied by the final state of that trajectory. Since $\mathcal{G}$ must be evaluated at the end of the trajectory we also sometimes proceed as if $\mathcal{G}$ is of the form `at end` ϕ.

3 The Agent Programming Language GOAL

The agent programming language GOAL is a language for programming rational agents [10,4]. It provides constructs for specifying the *beliefs* and *goals* of

agents and for programming a *strategy for action selection*. GOAL agents derive their choice of action from their beliefs and goals. GOAL defines a programming *framework* rather than a concrete programming *language* because GOAL does not commit to any particular knowledge representation and may be combined with various knowledge representation languages such as Prolog, ASP, OWL, etc. The current implementation of GOAL is based on Prolog.

3.1 GOAL Framework

We assume that some *knowledge representation technology* is available that agents use to represent, reason, and update their beliefs and goals.

Definition 8. *(Knowledge Representation Technology)*
A knowledge representation technology (KRT) *is a triple* $\langle \mathcal{L}, \models, \oplus \rangle$ *with* $\mathcal{L}$ *a* language *to represent an agent's beliefs and goals,* $\models \subseteq 2^{\mathcal{L}} \times \mathcal{L}$ *a* consequence relation *for* $\mathcal{L}$*, and* $\oplus : 2^{\mathcal{L}} \times \mathcal{L} \mapsto 2^{\mathcal{L}}$ *an* update operator *that defines how a set of formulae is updated with a given formula. We assume that falsity* $\bot \in \mathcal{L}$*.*

A KRT $\langle \mathcal{L}, \models, \oplus \rangle$is a *plugin* for a GOAL framework. A second plugin component of a GOAL framework is a *set of actions Act* that GOAL agents may perform, typically dependent on the environment of the agents. A GOAL framework is abstractly defined first and then each component of a framework is explained.

Definition 9. *(*GOAL *Framework)*
A GOAL framework *based on a KRT* $\langle \mathcal{L}, \models, \oplus \rangle$ *is a tuple* $\langle \Psi, \mathcal{L}_\Psi, \models_\Psi, \mathcal{M} \rangle$ *where:*

- $\Psi \subseteq \mathcal{L} \times \mathcal{L} \times \mathcal{L}$ *is the set of possible* mental states *of* GOAL *agents,*
- $\mathcal{L}_\Psi$ *is a language of* mental state conditions*,*
- $\models_\Psi \subseteq \Psi \times \mathcal{L}_\Psi$ *defines the truth conditions of mental state conditions, and*
- $\mathcal{M} : \Psi \times Act \rightharpoonup \Psi$ *is a* mental state transformer*.*

A *mental state* $m \in \Psi$ is a triple $m = \langle K, \Sigma, \Gamma \rangle$ with $K, \Sigma, \Gamma \subseteq \mathcal{L}$ which consists of a *knowledge base* K, *belief base* Σ, and *goal base* Γ. The knowledge base of a GOAL agent consists of *static conceptual or domain knowledge* that does not change. This means that performing an action does not modify a knowledge base and applying the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$ is constrained such that $K = K'$. The belief base consists of the beliefs of an agent that may change due to actions that are performed. Assuming that φ represents the effects of performing action α, the belief update operator $\oplus$ is used to compute the new set of beliefs and applying the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$ is constrained such that $\Sigma' = \Sigma \oplus \varphi$. The goal base consists of the goals of an agent, e.g. to have one block on top of another (sometime in the future). Typically, additional *rationality constraints* are imposed on mental states, such as that K, Σ, and Γ are consistent (i.e. $\bot$ does not follow from any of these sets). The language $\mathcal{L}_\Psi$ of *mental state conditions* is used by GOAL agents to inspect their mental state. It consists of *mental atoms* $\mathbf{B}(\varphi)$, to inspect an agent's beliefs, and $\mathbf{G}(\varphi)$, to inspect an agent's goals, where

$\varphi \in \mathcal{L}$, and combinations of such atoms by means of Boolean operators. It is not allowed to nest the operators **B** and **G**. The semantics $\models_\Psi$ of mental state conditions is derived from the consequence relation $\models$ provided by the KRT, and is defined as follows (the Boolean operators are defined as usual):

$$\langle K, \Sigma, \Gamma \rangle \models_\Psi \mathbf{B}(\varphi) \text{ iff } K \cup \Sigma \models \varphi$$
$$\langle K, \Sigma, \Gamma \rangle \models_\Psi \mathbf{G}(\varphi) \text{ iff } K \cup \Gamma \models \varphi$$

This definition also clarifies the distinct role of knowledge and beliefs. Conceptual knowledge may be used *in combination both with beliefs and goals*, which allows e.g. an agent to conclude that it wants to put block *a above* block *c* if it wants block *a* on top of *b* on top of *c* using a rule that defines the concept *above*.

The actions *Act* that GOAL agents may perform are specified as STRIPS-style triples $\langle \alpha(\boldsymbol{x}), \varphi, \varphi' \rangle$ where α is the name of the action - including parameters $\boldsymbol{x}$, $\varphi \in \mathcal{L}$ is the action's *precondition*, and $\varphi' \in \mathcal{L}$ is the action's *postcondition*. Multiple action specifications for the same action α can be provided, allowing for non-deterministic actions. An action is said to be *enabled* when its precondition holds. Agents do not have direct access to their environment and have to inspect their mental state (beliefs) to verify that an action is enabled. A GOAL agent makes a choice which action it will perform from the possibly multiple actions that are enabled by a *rule-based action selection mechanism* that uses so-called *action rules*. Action rules are of the form **if** ψ **then** α and define a strategy or policy for action selection of an agent. Here, ψ is a mental state condition that specifies when action α may be selected. If ψ follows from the agent's current mental state, we say that α is *applicable*. Finally, if an action is both applicable and enabled, it is said to be an *option*, one of which is non-deterministically choosen by an agent for execution.

GOAL agents thus maintain a mental state and derive their choice of action from their beliefs and goals. A GOAL agent consists of the *initial beliefs and goals*, *specifies the preconditions and effects of the actions* available to the agent, and contains a set of *action rules* to select actions for execution at runtime. Like the knowledge base, the action specifications and action rules are static.

Definition 10. *(GOAL Agent)*
A GOAL agent *is a tuple* $\langle K, \Sigma, \Gamma, R, A \rangle$ *where* $\langle K, \Sigma, \Gamma \rangle$ *is a mental state,* R *is a set of action rules* **if** ψ **then** α*, and* A *is a set of action specifications.*

Given the definition of a GOAL agent and the semantics of mental state conditions by $\models_\Psi$, we can define the notion of a *computation step* in which a GOAL agent performs an action.

Definition 11. *(Computation Steps)*
Let $Agt = \langle K, \Sigma, \Gamma, R, A \rangle$ *be a* GOAL *agent with a mental state* $m = \langle K, \Sigma, \Gamma \rangle$*. Then the set of possible* computation steps *that Agt can perform from* $\langle K, \Sigma, \Gamma \rangle$ *is denoted by* $\longrightarrow$ *and defined by:*

$$\frac{\textbf{if } \psi \textbf{ then } \alpha \in R, \langle \alpha, \varphi, \varphi' \rangle \in A, m \models_\Psi \psi \wedge \mathbf{B}(\varphi)}{m \xrightarrow{\alpha} \mathcal{M}(m, \alpha)}$$

This semantics allows for non-determinism in two ways. First, if multiple actions are options, one of these actions is non-deterministically choosen. Second, if one and the same action has multiple action specifications that can be executed simultaneously, one of these specifications is choosen non-deterministically. The latter allows for non-deterministic actions such as throwing a dice.

The action semantics of GOAL induces a set of possible *computations*. A computation is defined as an infinite sequence of mental states m_i and actions α_i, such that mental state m_{i+1} is obtained from m_i by applying the transition rule of Definition 11 with action α_i. Although computations are infinite, intuitively, the actions of a finite prefix of a computation that achieve the agent's goals may be viewed as a plan that a planner may return to achieve these goals. As GOAL agents are non-deterministic, the semantics of a GOAL agent is defined as a *set* of possible computations that start in the agent's initial mental state. A GOAL agent thus may be viewed as defining a *plan search space*; below, we make this statement precise and formally show GOAL can be used as a planning formalism.

Definition 12. *(Run, Meaning of a* GOAL *Agent)*
A run or computation r *is an infinite sequence* $m_0, \alpha_0, m_1, \alpha_1, \ldots$ *of mental states* m_i *and actions* α_i *such that* $m_i \xrightarrow{\alpha_i} m_{i+1}$*, or for all* α*:* $m_i \not\xrightarrow{\alpha}$ *and* $m_j = m_i$ *for all* $j > i$ *and* $\alpha_j = \mathbf{skip}$ *for all* $j \geq i$*.*

We write r_i^m *to denote the mental state at point* i *in* r *and* r_i^a *to denote the action performed at point* i *in* r*. The meaning* $\mathcal{R}_{Agt}$ *of a* GOAL *agent named Agt with initial mental state* m_0 *is the set of all runs starting in that state.*

3.2 Temporal GOAL

The instantiation of a GOAL framework with linear temporal logic as KRT is called *temporal* GOAL. Temporal GOAL has been introduced in [10] but here we use a first-order variant and show how planning problems with temporal planning constraints can be embedded in GOAL. The KRT plugin of temporal GOAL is $\langle \mathcal{L}_{LTL}, \models_{LTL}, \oplus \rangle$ where $\mathcal{L}_{LTL}$ is a first-order linear temporal language and $\models_{LTL}$ is the usual consequence relation associated with $\mathcal{L}_{LTL}$ [7]. The standard language of linear temporal logic is extended with two special predicates $\mathbf{do}(\alpha)$ with $\alpha \in Act$ and *fail*, where $\mathbf{do}(\alpha)$ means that action α is performed and *fail* indicates a failure to achieve a goal. Formally, $\mathcal{L}_{LTL}$ is defined as an extension of $\mathcal{L}_0$ with temporal operators to facilitate the compilation of PDDL to GOAL.

Definition 13. *(Linear Temporal Logic)*
The language $\mathcal{L}_{LTL}$*, with typical element* χ*, is defined by:*

$$\chi ::= \top \mid \mathit{fail} \mid (\phi \in \mathcal{L}_0) \mid \mathbf{do}(\alpha \in Act) \mid \neg\chi \mid \chi \wedge \chi \mid \forall(x \in \mathcal{V}).\chi \mid \bigcirc\chi \mid \chi \ \mathbf{until}\ \chi$$

$\Diamond\chi$ and $\Box\chi$ are the usual abbreviations and we use χ **before** χ' as abbreviation for $\neg(\neg\chi \ \mathbf{until}\ \chi') \wedge \Diamond\chi'$. Temporal GOAL uses an encoding of action specifications or planning operators into $\mathcal{L}_{LTL}$ as in [5,10,11]. This allows us to incorporate a *declarative* encoding of action preconditions and effects in the

knowledge base of a GOAL agent. Briefly, preconditions π_α are mapped onto *precondition axioms* of the form $\Box(\mathbf{do}(\alpha) \rightarrow \pi_\alpha)$ expressing that action α may be performed only if its precondition π_α holds. Effects are represented by a temporal encoding of successor state axioms. *Successor state axioms* are of the form $\Box(\bigcirc p(\boldsymbol{x}) \leftrightarrow (A_p^+ \vee (p(\boldsymbol{x}) \wedge \neg A_p^-)))$ with A_p^+ and A_p^- disjunctions of the form $\mathbf{do}(\alpha_1) \vee \ldots \vee \mathbf{do}(\alpha_m)$. Here, A_p^+ collects all actions that have $p(\boldsymbol{x})$ as effect and A_p^- all actions that have $\neg p(\boldsymbol{x})$ as effect. Intuitively, a successor state axiom expresses that $p(\boldsymbol{x})$ is the case in the next state iff an action is performed that has $p(\boldsymbol{x})$ as effect, or $p(\boldsymbol{x})$ is the case and no action with $\neg p(\boldsymbol{x})$ as effect is performed. To account for conditional effects a small modification is needed: If $(\neg)p(\boldsymbol{x})$ is an effect of α conditional on ϕ, then replace $\mathbf{do}(\alpha)$ by $\mathbf{do}(\alpha) \wedge \phi$ [5].

Now we are ready to define the components $\langle \Psi, \mathcal{L}_\Psi, \models_\Psi, \mathcal{M} \rangle$ of temporal GOAL. $\mathcal{L}_\Psi$ and $\models_\Psi$ are as defined above, given that $\mathcal{L} = \mathcal{L}_{LTL}$. $\mathcal{L}_\Psi$ thus allows temporal formulae inside the scope of the $\mathbf{B}$ and $\mathbf{G}$ operators, and we can use this to define several common sense notions of goals (see also [10]). Let $\mathbf{goal}\,\chi$ be short for $\mathbf{G}\chi \wedge \neg\mathbf{B}\chi$. $\mathbf{goal}\,\chi$ holds if χ is a goal and is not believed to occur inevitably, and corresponds more closely to the intuitive notion of a goal as being something that requires effort. When χ is of the form $\Diamond\chi'$ we say χ is an *achievement goal*. Note that we have $\mathbf{B}\chi \rightarrow \mathbf{G}\chi$ due to the rationality constraint $\Sigma \subseteq \Gamma$, which implies *realism* [6], but we do not have $\mathbf{B}\chi \rightarrow \mathbf{goal}\,\chi$.

The set of mental states Ψ is restricted such that: knowledge bases only consist of PDDL axioms and precondition and frame axioms as defined above, belief bases are sets of literals, and goal bases are sets of temporal logic formulae. A rationality constraint is imposed on mental states $\langle K, \Sigma, \Gamma \rangle$ such that $(K \cup \Sigma) \subseteq \Gamma$ (cf. [10]). There are various reasons for this constraint, both conceptually as well as technically, but due to space restrictions we refer to [10] and only make two brief remarks here: The constraint implies that (i) K, Σ and Γ are mutually consistent, i.e., $K \cup \Sigma \cup \Gamma \not\models \bot$, which means that the agent cannot have something as a goal that is never realizable according to its beliefs, and (ii) statements believed to be currently the case are part of the goal base, which allows us to use the standard definition of progression of $\mathcal{L}_{LTL}$ formulae [2].

What remains is that we need to provide a definition of the mental state transformer $\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K', \Sigma', \Gamma' \rangle$, i.e. we must specify how Σ' and Γ' can be obtained when α is performed (recall that knowledge bases do not change). We first specify how the belief base Σ' is obtained. To this end, the effects of performing α are collected in a set with *positive effects* $\mathit{Eff}^+_{LTL} = \{p(\bar{\mathbf{c}}) \mid K \cup \Sigma \cup \{\mathbf{do}(\alpha)\} \models \bigcirc p(\bar{\mathbf{c}})\}$ and a set with *negative effects* $\mathit{Eff}^-_{LTL} = \{\neg p(\bar{\mathbf{c}}) \mid K \cup \Sigma \cup \{\mathbf{do}(\alpha)\} \models \bigcirc \neg p(\bar{\mathbf{c}})\}$. Using these sets, and by slightly abusing notiation, we define:

$$\Sigma' = \Sigma \oplus (\mathit{Eff}^+_{LTL} \wedge \mathit{Eff}^-_{LTL}) \stackrel{df}{=} \mathit{Eff}^-_{LTL} \cup \mathit{Eff}^+_{LTL}$$

Finally, we need to specify Γ'. To do so, we use the progression operator from [2] *relative to the current belief base* Σ, but only specify some clauses of the inductive definition due to space limitations. For the base case $\phi \in \mathcal{L}_0$, $\mathit{Progress}(\phi, \Sigma) = \top$ if $\Sigma \models \phi$, otherwise $\mathit{Progress}(\phi, \Sigma) = \bot$. $\mathit{Progress}(\bigcirc\varphi, \Sigma) = \varphi$, the case

that requires the constraint $\Sigma \subseteq \Gamma$ to be in place as it allows that a goal ϕ is entailed by the agent's beliefs. $Progress(\forall x.\varphi, \Sigma) = \bigwedge_{c \in \mathcal{C}} Progress(\varphi[c/x], \Sigma)$. We assume that $\bot \vee \chi$ is reduced to χ, $\bot \wedge \chi$ to $\bot$, $\top \vee \chi$ to $\top$, etc. In order to ensure progression always yields a consistent goal base Γ', some syntactic restrictions are imposed on the temporal formulae allowed in goal bases: they need to be in negation normal form (negation occurs only in front of atoms) and may not have occurrences of disjunction $\vee$ or the next operator $\bigcirc$.

Progression of a formula may result in $\bot$ which indicates that one of the goals has not been achieved, and, consequently, that the actions selected cannot be viewed as a plan for achieving these goals. To record such failure the special predicate *fail* has been introduced above, and is used to replace $\bot$ which also restores consistency. That is, the new goal base denoted by $Progress^{fail}(\Gamma, \Sigma)$ after performing an action is given by the set $\bigcup_{\varphi \in \Gamma} Progress(\varphi, \Sigma)$ where $\bot$, if present, has been replaced by *fail*. Finally, we define:

$$\mathcal{M}(\langle K, \Sigma, \Gamma \rangle, \alpha) = \langle K, \Sigma', Progress^{fail}(\Gamma, \Sigma) \rangle$$

where $\Sigma' = \mathit{Eff}^{-}_{LTL} \cup \mathit{Eff}^{+}_{LTL}$ and Eff^{+}_{LTL} and Eff^{-}_{LTL} are defined as above.

4 Compiling PDDL Problems into GOAL Agents

In this Section we show that a GOAL framework is expressive enough to define a planning problem from $PDDL_{Ax+CE+PC}$, given that such a framework is instantiated with a temporal logic KRT.

To compile a PDDL planning problem into a GOAL agent we use the concept of a compilation scheme. A *compilation scheme* is a mapping $\mathbf{F}$ from planning problems Π to GOAL agents $\mathbf{F}(\Pi)$ such that: (i) there exists a plan for Π iff there exists a run of $\mathbf{F}(\Pi)$ that achieves all goals, (ii) the translations of the initial and goal states of Π can be computed in polynomial time, and (iii) the size of $\mathbf{F}(\Pi)$ is polynomial in the size of Π. Compilation schemes in our sense are similar to that used in [13] but for obvious reasons differ in that we map to a GOAL agent instead of a planning problem in e.g. a different fragment of PDDL.

The compilation scheme $\mathbf{f}$ defined below maps every problem instance Π of the PDDL fragment $PDDL_{Ax+CE+PC}$ to a GOAL agent $\mathbf{f}(\Pi)$. The scheme $\mathbf{f}$ is defined by a tuple of functions $\langle f_{ax}, f_{at}, f_c, f_r, f_{as}, t_i, t_g, t_{pc} \rangle$ that map different parts of a PDDL problem $\langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$ to corresponding GOAL agent $\langle K, \Sigma, \Gamma, R, A \rangle$ components.

Definition 14. *(Compiling PDDL Problems to* GOAL *Agents)*
Let $\Pi = \langle \Delta, \mathcal{C}, \mathcal{I}, \mathcal{G}, \mathcal{PC} \rangle$ *be a PDDL problem and* $\Delta = \langle \mathcal{L}_0, \mathcal{A}, \mathcal{O} \rangle$. *The scheme* $\mathbf{f}$ *from PDDL problems to* GOAL *agents* $\mathbf{f}(\Pi) = \langle K, \Sigma, \Gamma, R, A \rangle$ *is defined by:*

- $K = f_{ax}(\mathcal{A}) \cup f_{at}(\mathcal{O}) \cup f_c(\mathcal{C})$,
- $\Sigma = t_i(\mathcal{I}) = \hat{\mathcal{I}}$,
- $\Gamma = t_g(\mathcal{G}) \cup t_{pc}(\mathcal{PC})$,
- $R = f_r(\Delta)$,
- $A = f_{as}(\Delta)$

The scheme $\mathbf{f}$ maps PDDL axioms into the knowledge base K by applying the well-known completion operator used to compute the completion of logic programms to these axioms, i.e. $f_{ax}(\mathcal{A}) = comp(\mathcal{A})$. Intuitively, the completion $comp(\mathcal{A})$ replaces implications with equivalences (cf. [1]). f_{at} maps the operators $\mathcal{O}$ to an action theory in $\mathcal{L}_{LTL}$ as discussed above that is also part of K. Finally, a *domain closure axiom* $f_c(\mathcal{C}) = \forall x.(x = c_1 \vee \ldots \vee x = c_n)$ is added to K, where $c_1, \ldots, c_n$ exhaust the constants in $\mathcal{C}$ (cf. [13]). The fact that a knowledge base of a GOAL agent is static corresponds with the fact that a PDDL domain does not change over time. The function t_i maps the initial state $\mathcal{I}$ to the belief base, such that $t_i(\mathcal{I}) = \hat{\mathcal{I}} = \{p(\bar{\mathbf{c}}) \in \mathcal{I}\} \cup \{\neg p(\bar{\mathbf{c}}) \mid \mathcal{I} \not\models p(\bar{\mathbf{c}}), p(\bar{\mathbf{c}}) \in \mathcal{B}\}$. t_g maps the goal $\mathcal{G} = \texttt{at end}(\phi_g)$ to $\Diamond\phi_g$. Assuming that $\mathcal{G} = \texttt{at end}\phi_g$ is the main goal, t_{pc} maps the planning constraints $\mathcal{PC}$ to the goal base as follows:

- $t_{pc}(\forall x.\varphi) = \forall x.t_{pc}(\varphi)$
- $t_{pc}(\texttt{always}\ \phi) = \phi\ \textbf{until}\ \phi_g$
- $t_{pc}(\texttt{sometime}\ \phi) = \phi\ \textbf{before}\ \phi_g$
- $t_{pc}(\texttt{at-most-once}\ \phi) = \phi\ \textbf{before}\ \phi_g \rightarrow (\phi\ \textbf{until}\ (\neg\phi\ \textbf{until}\ \phi_g))$
- $t_{pc}(\texttt{sometime-after}\ \phi\ \phi') = \phi\ \textbf{before}\ \phi_g \rightarrow (\phi\ \textbf{before}\ (\phi'\ \textbf{before}\ \phi_g))$
- $t_{pc}(\texttt{sometime-before}\ \phi\ \phi') = \phi\ \textbf{before}\ \phi_g \rightarrow (\phi'\ \textbf{before}\ (\phi\ \textbf{before}\ \phi_g))$

This translation shows the difference between assuming a finite horizon as in planning that is not made in GOAL and has to be enforced by it. The function $f_r(\Delta)$ maps each of the actions $\alpha \in \mathcal{O}$ to an action rule **if** $\top$ **then** α in the program section, and f_{as} maps each operator definition $\langle \alpha, \pi_\alpha, \epsilon_\alpha \rangle \in \Delta$ to an action specification $\langle \alpha, \pi_\alpha, \top \rangle$. This works since effects are encoded in the action theories stored in the knowledge base.

Theorem 1. *There exists a solution for a PDDL problem Π iff there is a run r of the* GOAL *agent $\mathbf{f}(\Pi)$ such that for some i with $r_i^m = \langle K, \Sigma, \Gamma \rangle$: $\Sigma \models \Gamma$.*

Proof. Due to space limitations we only provide a sketch. The proof proceeds by first showing that the translation of the PDDL axioms and domain closure assumption is correct. Consecutively, we show that updates by performing actions in PDDL correspond with those derived from temporal action theories in GOAL. Finally, we show that there exists a plan (solution) iff the goal base of the compiled GOAL agent after performing the corresponding action sequence is empty and does not contain *fail*.

The theorem states that a planning problem can be represented by a GOAL agent. Another view on this result is that the meaning of the GOAL agent $F(\Pi)$ defines the *plan search space*. Obviously, part of this result is derived from the expressiveness of linear temporal logic, which allows to encode the semantics of planning operators and plan constraints. However, GOAL agents do not use temporal logic to select actions but use action rules to do so. A *plan search space* thus is defined by the GOAL framework component of temporal GOAL (more specifically, Definition 11) and not by the temporal logic plugin.[1]

[1] Another indication of this fact is that we do not need *compatibility axioms* as in [11].

It is clear that each of the functions $\langle f_{at}, f_c, f_r, f_{as}, t_i, t_g, t_{pc} \rangle$ that define $\mathbf{f}$ execute in polynomial time, and, as a consequence, the size of the GOAL agent obtained by applying $\mathbf{f}$ is polynomial in the size of the original PDDL problem.

Corollary 1. $\mathbf{f}$ *defines a compilation scheme.*

It is clear that GOAL agents that are obtained by mapping a PDDL problem into GOAL can be translated back into a PDDL problem. This is not the case in general, however. For example, it is not clear how to map multiple achievement goals of the form $\Diamond\phi$ into a PDDL problem. Similarly, it is not clear how to map non-trivial action rules that use mental atoms of the form $\mathbf{G}(\varphi)$ into a PDDL problem. The question is which restrictions need to be imposed on GOAL agents to be able to map them into a PDDL problem. We propose the following restrictions as a *sufficient* set, but it remains to establish a *necessary* set:

1. The knowledge base consists of a *stratified* axiom set, and an LTL action theory.
2. The belief base is a set of literals $\hat{S}$ where S is a PDDL state.
3. The goal base consists of a *single* achievement goal of the form $\Diamond\phi$ with $\phi \in \mathcal{L}_0$ and, possibly, additional deadline goals of the form φ **until** ϕ and φ **before** ϕ.
4. The program section consists of *reactive* action rules only, i.e. rules that have only occurrences of belief atoms of the form $\mathbf{B}(\phi)$ with $\phi \in \mathcal{L}_0$ in their conditions.
5. Variables that occur in preconditions are parameters of the corresponding action.
6. All specified actions specified in the GOAL agent are *deterministic*. (PDDL does not allow for non-deterministic actions.)

Item (3) highlights one of the differences between dynamic agents, that may have multiple goals and dynamically adopt and/or drop goals, and static planning tasks. Alternatively, planners look for a fixed horizon determined by the goal state which defines a temporal window to which plan constraints apply, whereas this window needs to be made explicit in the mapping to GOAL.

5 Conclusion

We have shown that GOAL can be used as a planning formalism. To this end, the notion of a GOAL framework has been introduced that may be instantiated with various knowledge representation technologies. A GOAL framework defines the structure and semantics of a GOAL agent and has been introduced to be able to make a distinction between the expressiveness provided by GOAL itself and that provided by a KRT plugin. Temporal GOAL, a GOAL framework with a linear temporal logic plugin, has been used to compile planning problems with planning constraints into GOAL agents using so-called compilation schemes in the sense of [13]. This paves the way for integrating a planner such as TLPlan [2] into GOAL and to combine the strenghts of planners with those of agent programming

languages. We plan to integrate a planner into GOAL in combination with the notion of a module introduced in [9]. The idea is to introduce a variant called a planmodule where the context condition is used to define when to call a planner.

As argued, the semantics of GOAL may be viewed as defining a plan search space. One very interesting avenue for future research is how GOAL action rules with non-trivial mental state conditions can be used to reduce this search space. The idea is similar to how Golog programs [13] may reduce the search space and how heuristic knowledge can be used in TLPlan [2] and PDK [11].

References

1. Apt, K.R., Bol, R.: Logic programming and negation: A survey. Journal of Logic Programming 19, 9–71 (1994)
2. Bacchus, F., Kabanza, F.: Using temporal logics to express search control knowledge for planning. Artificial Intelligence 16, 123–191 (2000)
3. Baier, J., McIlraith, S.: Planning with first-order temporally extended goals using heuristic search. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI 2006), Boston, MA, July 2006, pp. 788–795 (2006)
4. Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.): Multi-Agent Programming: Languages, Platforms and Applications. Springer, Heidelberg (2005)
5. Cerrito, S., Mayer, M.C.: Using Linear Temporal Logic to Model and Solve Planning Problems. In: Giunchiglia, F. (ed.) AIMSA 1998. LNCS (LNAI), vol. 1480, pp. 141–152. Springer, Heidelberg (1998)
6. Cohen, P.R., Levesque, H.J.: Intention Is Choice with Commitment. Artificial Intelligence 42, 213–261 (1990)
7. Emerson, E.A.: Temporal and Modal Mogic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science (1990)
8. Gerevini, A., Long, D.: Plan constraints and preferences in PDDL3. Technical report, Department of Electronics for Automation, University of Brescia (2005)
9. Hindriks, K.V.: Modules as Policy-Based Intentions. In: Dastani, M.M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 156–171. Springer, Heidelberg (2008)
10. Hindriks, K.V., van Riemsdijk, M.B., van der Hoek, W.: Agent programming with temporally extended goals. In: Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009 (2009)
11. Mayer, M.C., Limongelli, C., Orlandini, A., Poggioni, V.: Linear temporal logic as an executable semantics for planning languages. Journal of Logic, Language and Information 16 (2007)
12. Pednault, E.P.D.: ADL and the State-Transition Model of Action. Journal of Logic and Computation 4(5), 467–512 (1994)
13. Röger, G., Helmert, M., Nebel, B.: On the Relative Expressiveness of ADL and Golog. In: Proc. of the Eleventh Int. Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 544–550. AAAI Press, Menlo Park (2008)
14. Sardina, S., de Silva, L.P., Padgham, L.: Hierarchical planning in BDI agent programming languages. In: Proc. of the Fifth Int. Conference of Autonomous Agents and Multi-Agent Systems (AAMAS 2006), pp. 1001–1008 (2006)
15. Thiébaux, S., Hoffmann, J., Nebel, B.: In defense of PDDL axioms. Artificial Intelligence 168, 38–69 (2005)

Towards Pattern-Oriented Design of Agent-Based Simulation Models

Franziska Klügl and Lars Karlsson

School of Science and Technology,
Örebro University, Örebro, Sweden
{franziska.klugl,lars.karlsson}@oru.se

Abstract. The formalization and use of experiences in good model design would make an important contribution to increasing the efficiency of modeling as well as to supporting the knowledge transfer from experienced modelers to modeling novices. We propose to address this problem by providing a set of model design patterns inspired by patterns in Software Engineering for capturing the reusable essence of a solution to specific partial modeling problem. This contribution provides a first step formulating the vision and indicating how patterns and which types of patterns can play a role in agent-based model design.

1 Introduction

Since the seminal studies of Willemain [1] on how modeling experts actually build models, it is known that modeling and simulation processes are mostly guided by expert intuition.

Despite of its obviously intuitive way of modeling, one must admit that the development of a concept model and the design of an agent-based simulation from that is a very challenging task – especially for starting modelers. Finding the right abstractions, focussing on only the relevant relations and modeling on the right level of detail are issues that hardly an experienced modeler can handle well without following a painful trial and error procedure. For novices in modeling and simulation – with or without (software) engineering education –, this phase is particularly difficult and frustrating as they find no guidelines and heuristics for these early, yet decisive phases of a simulation study. Engineering processes advising the different steps in a systematic simulation study are not helpful in these early phases of modeling. Unfortunately, the greatest advantage of multi-agent simulation - its high degree of freedom in model design - turns out to be a curse as there are no constraints and restrictions on what can be formulated in a model. Therefore, especially novices may feel lost in the wide field of alternatives.

The problem of guiding the development of a model concept can hardly be solved from a general methodological point of view as it involves too much domain-specific knowledge. However, we can formulate iterative strategies for model development in general and support the model design on a more technical

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 41–53, 2009.

level. This contribution is part of larger vision of advancing the methodological status of agent-based simulation towards systematic engineering starting from a model concept to the analysis of the simulation results.

Our idea for supporting technical model design bases on the well-known concept of software design patterns that – since their first introduction by Gamma et al., [2] – became a success story in object-oriented software design. Design patterns are meanwhile used in undergraduate courses in computer science for teaching proper software design. They are acknowledged as an efficient vehicle for transporting knowledge and experiences about how to solve particular problems. Therefore the idea of transferring the concept to modeling and simulation seems to be promising - especially for the case of agent-based simulation. While patterns address good simulation model design, components for modeling as proposed in [3] are provided as implemented, configurable building blocks. Both concepts address model reuse, but they are clearly different.

The next section gives the state of art in pattern-oriented design of agent-based software, but also examines whether there are similar concepts of patterns in simulation and modeling. This is followed by a section on patterns for agent-based simulation tackling general representation and different categories and examples. The contribution ends with a short conclusion and indication of future work.

2 Pattern-Oriented Modeling

About ten years ago, software patterns became popular [2]. The basic idea behind those was to capture experience about good design for particular problems in a uniform and schematic way for supporting its reuse. A pattern can be seen therefore as a recurrent problem and its solution on conceptual, architectural or design level. Thus, the design of software should be ideally systemized to analysis and identification of basic problems item to solve, followed by the selection of appropriate patterns and instantiation and adaption to the particular case.

2.1 Patterns in the Agent World

Due to the obvious usefulness of patterns for software design, it is not surprising that soon patterns for agent-based software – not simulation – have been discussed and defined. Weiss [4] gives a good introduction to how to develop and use patterns in the development of agent-based software. Oluyomi and others [5] survey and analyze existing patterns for agent-oriented software and classify many of them. This work is accompanied by second publication from the same group [6]. There, the authors systematically deal with description templates appropriate for multi-agent systems with the aim of making different specifications of similar or equal patterns more comparable. In contrast to this, Sauvage [7] tackles more a meta-level view on patterns introducing the categories of anti pattern, metaphors and meta pattern.

Kendall and others [8] give patterns for several layers of an agent society such as sensory layer, action layer, collaboration and mobility layers. The patterns

range from *The Layered Agent Pattern* to patterns for different types of agents with different abilities with respect to agent capabilities as sensory-level patterns; *intention pattern* or *prioritizer pattern* as action-level patterns; agent system patterns ranging from *conversation pattern* to *facilitator* and *proxy pattern* and finally *clone* or *remote configurator* as examples for mobility patterns. They hereby use a uniform schema consisting of problem, forces, solution, variations and known uses.

Aridor and Lange [9] deal with patterns for mobile agents classified as task pattern, like *Master-Slave*, as interaction pattern, like *Meeting* or *Messenger* and as travel pattern, e.g. *itinerary* for agent routing. More patterns for itineraries can be found in the work of Chacon et al., [10] – whereas they mean more behavioral patterns – like the *Sentinel Agent Behavior Pattern* for persistent monitoring, than a routing pattern as in the previous case. Social patterns that help to design interactions between agents can be found in a publication by Do et al., [11]. They categorize social patterns into peer patterns such as the *booking pattern* or the *call-for-proposal* pattern and into mediation patterns such as *monitor pattern* or *broker pattern*. Only the *booking pattern* is fully elaborated until code generation in this publication. A list of similar mediation patterns is given in a paper by Hayden et al., [12]. More recently, one can find patterns for specific application domains, mostly in the area of information agents. A very extensive list of patterns can be found in the PhD thesis of Oluyomi [13].

For agent-based simulation, these patterns have to carefully evaluated and potentially revised. The main reason lies in the general differences between agent-based software and agent-based simulation: The constraints for the design of the simulated multi-agent system are much higher as it has to credibly correspond to an original system. Whereas functional and technical requirements of a software allow more alternatives in appropriate design, the main directive for simulation model design is its validity. Hereby, structural validity refers to the correspondence between the design of the model and the real-world system [14]. Thus, technical patterns, such as yellow pages pattern are not relevant for multi-agent simulations, as long as they don't need to be part of the model due to the objective of the simulation study. Additional patterns are relevant due to the relevance of the environment and of particular feedback loops hopefully leading to some given macroscopic behavior - for example in terms of population dynamics. This is the most difficult aspect in model design and is therefore the most attractive for formalizing experiences in terms of design pattern.

2.2 "Patterns" in the Simulation World

Grimm and Railsback [15] introduced so called pattern-oriented modeling for individual- respectively agent-based simulation in ecology. They thereby refer to patterns in given data that are to be reproduced by the simulation. Based on data patterns, a systematic model development procedure is defined. This is a different form of pattern than we use it here where we are discussing design patterns that should support the production of a well-structured model that resembles all required data patterns or stylized facts.

Similar pattern-like building blocks also have been established in mathematical, equation-based modeling, especially in biological simulation. The researcher analyzes what kind of relationship may be assumed between two or more variables and then selects the appropriate mathematical function from to a catalogue of well known formula. Finally, she or he fits the parameters of the functions to available data. These functions can be seen as basic model patterns. Examples are the exponential growth function or saturation curves. In his book on biological modeling, Haefner [16] lists a complete toolbox of functions for modeling different relationships that are usually applied in modeling and simulation in the area of population biology. Also the basic Lotka-Volterra equations of a predator prey system form elements of such a model schemata library. In [17] a pattern language for spatial process models is given; the examples are more like full model description. Unfortunately, such function patterns cannot be transferred to agent-based simulation as – in contrast to these models – agent-based models *generate* macroscopic dynamics from the agents behavior whereas the macroscopic models *describe* the relations and dynamics on the aggregate level. Nevertheless, it would be a most interesting – and most challenging – form of design support to find agent population patterns that produce the corresponding dynamics on the macroscopic level.

Also, in object-oriented simulation, pattern-based methods for developing simulators were proposed [18]. Patterns are hereby used for putting together objects or components that form the building blocks for a model. Using code templates associated with a pattern, the code for a complete simulator could be instantiated. Such a system was developed for the domain of building simulation.

3 Agent-Based Simulation Model Design Patterns

Considering agent-based modeling and simulation, the actual level of patterns may not always be obvious: the technical, detailed implementation-near level and a more conceptual design level. An example for the former is the *configuration interface pattern* describing how to aggregate all start values into one interface object. Examples for the latter are patterns of *limited exponential growth* or a *food web pattern*. One may further divide model design patterns into agent architecture patterns that describe appropriate ways of conceptualizing an agent itself, patterns for agent models generating certain population dynamics, and interaction patterns for capturing dynamics among agents and between agents and their environment. Also, meta-level patterns are possible that describe how different design pattern could be combined. As with software engineering patterns, all of these may seem trivial for the experienced multi-agent simulation developer, but may be highly valuable for starters, but also for general characterization of possible agent models.

In the following we will discuss several pattern categories and patterns in more or less detail; this is not meant as a complete list but more like an indication of what can be possible. First, we will set the general frame for approaching such a pattern by giving a schema.

3.1 Pattern Schema

Many experienced modeler implicitly use patterns of agent behavior. In an analogous way to software design patterns, model design patterns may be made explicit using some predefined scheme. Thus, before going into the details of particular model design patterns, we have to discuss a proper scheme for making the experiences explicit and thus reusable. As mentioned above, there are many suggestions for schemes. The following is clearly inspired by the original pattern language [2].

Name: Each pattern needs an memorable name for referring to the pattern in a short yet descriptive way.
Intent: What is the problem that this pattern aims at solving? What kind of pattern/relations should be reproduced by it?
Scenario: In what situations or basic model configurations, does it make sense to apply this pattern?
Description: Short description of the functionality produced by the pattern.
Dependencies: Does the pattern only make sense in combination with other patterns? Do we need a specific form of data for reasonably handling the structures?
Agent System Structure: This is the actual content of the pattern. What agents classes are involved? How do they interact? What environmental structures are part of the pattern?
Agent Behavior: Exact specification of the agent behavior pattern. This corresponds to the original "code" attribute of a pattern scheme.
Technical Issues: Sometimes additional low-level technical aspects are important. This serves to prevent artifacts coming from poor implementation.
Example: If necessary, an additional example can be given for clarifying the usage of the pattern. In what models was the pattern successfully applied?
Configuration: For evaluating the properties of the pattern in the simulation context, the existence and effects of parameters have to be discussed.
Consequence: Using this pattern may have consequences for further design decisions. Information about this should be given here. This item is different from the original one: the original "consequences" were split up into configuration and consequences.
Related Patterns: Pointer to other patterns that may compete with this pattern or have similar properties.

This is a first attempt for finding a systematics for documenting agent-based simulation model design. In the following a number of patterns are sketched and example patterns are discussed in more detail using the above introduced pattern description scheme. We have identified the following pattern categories: agent architecture patterns, agent population patterns, interaction patterns and environmental patterns.

3.2 Agent Architecture Patterns

The first category that comes to mind is agent architectures. They form basic patterns for how to organize different modules and processes within an agent. A

model design pattern on the level of an agent architecture does not necessarily tackle full agent architectures, but also certain component interactions.

Agent architectures were in the focus of research from the beginning of Distributed Artificial Intelligence. Thus, one can already find a wealth of architecture suggestions in the literature with different complexities, intended applications, etc. During the last years the discussion about appropriate agent architectures has calmed down a bit. For agent-based simulation, the selection/design of an agent architecture can be difficult as the used generic architecture must be explicitly treated as a basic assumption that also has a correspondence to the original system or its influence on the results should be credibly justified.

Formulating agent architectures as patterns is quite popular in agent-oriented software design (see Section 2.1). As one can find psychological theories for some architectures, especially layered and BDI architectures, it is absolutely justified to add these full architectures to a list of agent model design patterns. But also smaller parts or architectural details can be interesting to be formulated as pattern. In the following we give an example of an agent architecture pattern that turned out useful in several applications: *Perception Memory Pattern*. It combines a particular form of knowledge representation with specific reasoning. Instead of providing a complete architecture, it may also be combined with other patterns.

Name: *Perception Memory Pattern*

Intent: The pattern shows how perception can be dealt with separately from interpretation. This increases efficiency as perception is appropriately buffered instead of allowing environmental scans whenever information is needed.

Scenario: This pattern makes sense when the agent needs information about the local environment more than once in its behavior specification. The general background is that memory space is cheaper than scanning the environment.

Description: The agent status contains a set of variables for memorizing perceptions. The first step in each update cycle consists in the agent scanning its environment and memorizing all noticed objects in the variable. A cascade of filters provides information necessary in the current situation based on the initial perception. The agent may always access the initially perceived information, instead of scanning the environment again. The pre-processed information is then used for decision making or for determining the agent's behavior.

Dependencies: This is a basic pattern that may be used in a variety of models with other diverse patterns built upon it.

Agent System Structure: The agent needs at least one variable or memory location for saving the memorized information about its surroundings: PerceptionMemory. Additional data structures contain preprocessed information

Agent Behavior: The behavior or an agent contains the following partial steps:

1. Initial Scan → PerceptionMemory
2. Filter PerceptionMemory appropriately → ProcessedPerception

3. Follow behavioral rules, use ProcessedPerception as if it would come from direct access in the environment if necessary go back to the second step.

Technical Issues: It must be secured that the basic scan happens at the appropriate point of time in an update cycle and sufficiently often.

This separation between basic environmental scan with memorizing all information that might be useful in the later context may also be used for implementing virtual parallelism where all agents sense the environment and in a second step process the sensed information.

Example: We applied this patter in all pedestrian simulation models. The agents scanned their complete environment within their range of perception and saved all objects in a *PerceptionMemory* variable. In the next step all objects that are recognized as obstacles are filtered and stored separately for being used in the collision avoidance rules. A second independent preprocessing step allows the agent to sort out whether it has already reached its goal or not.

Configuration: The most basic parameter is the range of initial scan (a distance in metric space or a number of hops in a graph environment). This initial scan must be done with sufficient distance as the environment is only scanned once and all eventualities have to be foreseen.

Consequence: -

Related Patterns: This pattern might be used in combination with different memory structure patterns, such as the *Mental Map Pattern*, etc.

Although it takes quite some space to describe the *Perception Memory Pattern*, it is basically trivial. However our experience showed that especially beginners again and again access the environment in time intervals where no change can happen just because they want to save memory. However depending on the particular implementation of perception, accessing the environment often is the most expensive operation.

Beside other, already mentioned patterns, we might indeed find it useful to specify a BDI architecture in terms of a pattern. A pattern formalizing how to best design an adaptive discrete choice in term of a *Discrete Choice With Feedback Pattern* may be useful when agents have to choose from a set of options.

3.3 Agent Population Patterns

Especially interesting are design patterns that capture agent control aspects for producing some specific overall behavior on the aggregate level, such as a given population dynamics. Similar to the set of useful functions in the book of Haefner [16], one may identify several agent-level behavior modules that may lead to specific relations on the macro level. Such patterns are useful for all forms of systematic design – bottom-up or top-down. It is clear that for a final model these mostly isolated patterns have to be integrated into environmental dynamics, other behavioral feedback mechanisms, etc. One can image a lot of different population dynamics generated by different behavior and interaction

behaviors on the agent level. The following *Exponential Growth Pattern* can be seen as the one of the simplest forms of such an agent population pattern.

Name: *Exponential Growth Pattern*
Intent: A population of agents should exhibit basic exponential growth in population numbers. This is basically the purest agent-based implementation of an exponential growth function.
Scenario: Useful for different scenarios where exponential growth of an agent population is necessary.
Description: Agent duplicate themselves. Duplication is triggered with a given individual probability.
Dependencies: none
Agent System Structure: There is just one agent class with one attribute. An additional global container may store references to all agents and serve as a bookkeeping device for the number of agents.
Agent Behavior: see figure 1

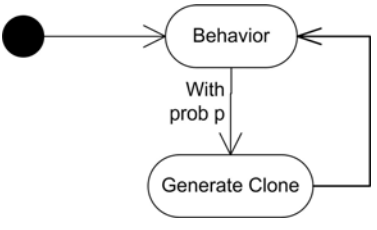


Fig. 1. Behavior specification for the agent behavior producing exponential growth

Configuration: There is only one parameter per agent, namely the duplication probability. The basic question is how to set this local agent parameter for producing the corresponding macro behavior. Basically the macro rate should equal the probability for duplication. However, this is not so simple as illustrated in figure 2.

Even without parameter variations, the outcome of a short simulation generated with the same initial conditions, varies a lot. This is due to the high degree of randomness in this model formulation.

A technical issue concerns the integration of new agents into the update schedule of the simulation. One has to pay attention, whether there are delays originating from the inclusion of new agents into the overall update sequence.

Example: This pure agent system pattern is completely unrealistic in reality as there is no unconstraint, unbounded growth. However, an exponential growth model might set the frame for a more complex one, modifying the reproduction probability.
Consequence: Its exponential growth: depending on the parameter the population growth tremendously fast, sufficient computational resources are necessary. The system easily gets out of reasonable population sizes. Due to

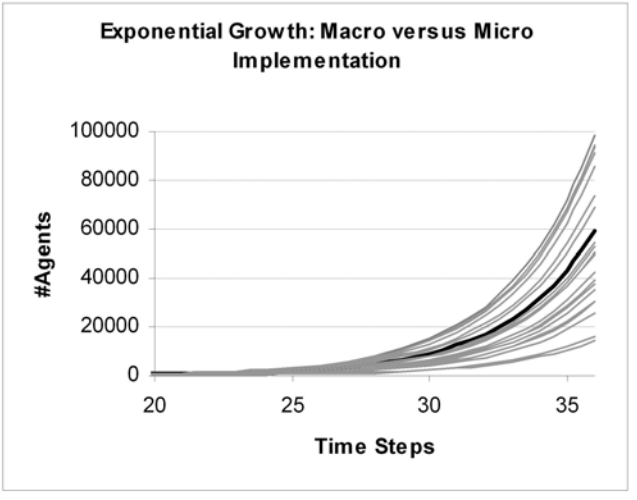


Fig. 2. Different micro runs do only in average resemble the macro model (black)

the relevance of the random component with direct influence on population sizes, it is hard to control.

Extensions: There are many obvious extensions addressing the decision for duplication: the probability may be replaced by a counter for some regular, deterministic reproduction. Both probability and counter-based reproduction can be modified by resource or energy bounds. Pattern-like structures can also be formulated for sexual reproduction, local density dependent duplication, etc.

Related Patterns: Pattern that remove agents from the simulated environment, such as age-dependent death, starvation or predation.

This *Exponential Growth Pattern* seems to be trivial, but it is also something that many modelers have used in their models without really reflecting about that this could be pattern – a special building block for a model that could be used to document best practice. As mentioned in the extensions section, there a set of potential variations. For bounded growth in interaction with other types of agents something like *n-species food web pattern* could be specified that tackles interacting populations of agents, but defined from an agent-perspective. In contrast to the following category the patterns here aim at reproducing certain macro-level population dynamics; the following focus on interaction between agents for coordination or for (self-)organization.

3.4 Interaction Patterns

In a similar way, one can specify good solutions to standard problems concerning negotiation among and organization of agents. Here, the intersection with agent-oriented software engineering patterns should be high - considering bidding, call-for-proposals, or even mediation pattern (see Section 2.1). As with the

agent architecture pattern, the degrees of freedom in design are constrained by the required correspondence between original and model. Additional patterns can be interesting besides the patterns suggested by the agent software community, such as the *Tag-based Interaction Pattern* or *Memory-based Interaction* that can be often found in models of iterated games. In biological as well as social science models a pattern can be found describing how some form of interaction leaves marks in the beliefs/memory of an agent that influences the selection of future interaction partners. This pattern may be denoted *Emergent Organization Pattern*. Interesting patterns may also be found for specific organization or relation networks – a pattern describing the initialization of a small world network is definitely useful.

3.5 Environment Patterns

Explicit space often plays an important role in agent-based simulation models. In most cases one can observe that space representation and consequently, environment-agent interactions are handled in similar ways. Patterns can be formulated like the *Background Cellular Automata Pattern* describing discrete cells that carry (dynamic) information that is perceived and modified not only by the agents, but also by the cells themselves potentially in relation to the values of their neighbor cells. Application examples can be found in certain pedestrian simulations like in the work of Bandini and co-workers [19] where a cellular automata is used to store the potential gradient for guiding the agents movement or in the well-known Sugarscape [20] model to represent the heterogenous renewable resources. Such a pattern would document how a cellular automata and its particular update could be used as the agents environment. Hereby a cell forms the position of an agent that accesses the state variable of that cell. A similar pattern for the continuous case may be something like a *Potential Field Pattern*.

Also for agent movement in space, patterns can be usefully applied. Just think about, how often a random walk has to be designed and how often the modeler starts anew thinking about whether a completely random change in direction from time to time or a slight random change in direction before every step is more appropriate. A *Random Walk Pattern* is a good way to formulate the experiences and guide future random walk designs. Similar considerations may result in a *Pheromone-based interaction* as a specific recurrent form of stigmergic interaction. Also more elaborate walking patterns can be useful, e.g. a *Path in Continuous Space Pattern* where the problem of obstacle avoidance in a continuous space without additional spatial structure is tackled.

4 Future Challenges: Formalization and Evaluation

The next steps in actually making these ideas viable would be a thorough examination of as many available models as possible for candidates of patterns. Those patterns then must be fully and precisely described and more clearly classified than we did in this paper. However, this raises the question what languages

should be used for precisely characterizing the patterns, respectively the different elements of their description. Original UML [21] and its current versions or a selection from the many agent-oriented extensions of UML (such as MESSAGE/UML [22]) can be seen as good candidates at least for partial aspects of the overall description.

One can see that the number of potentially useful agent-based model design patterns for supporting the design of the a multi-agent model can be quite long. We gave a few examples and indicated a number of others whose usefulness is immediately clear to somebody who was confronted with such a problem. Nevertheless the patterns in this list have to be evaluated, e.g. its usefulness and useability have to be tested.

What constitutes a good design pattern is ultimately an empirical question, and can only be answered by investigating what the outcome is when people actually use it. The usefulness of design patterns in software engineering have been empirically evaluated in a number of studies. The first study of that kind was conducted by Prechelt et al [23], who experimentally compared software maintenance with and without design patterns. They came to the conclusion that "pattern-relevant maintenance tasks were completed faster or with fewer errors if redundant design pattern information was provided." A similar study by Vokác [24] found that certain design patterns were easier to use than others, and resulted in considerable lower defect rates.

5 Conclusion

The process from conceptual to implemented model is the hardest part in modeling and simulation – especially when considering multi-agent simulations with their unlimited freedom of design. Therefore it is important to provide less experienced modelers with some guideline for a good model design. Following the success of patterns in object-oriented software engineering and consequently in agent-oriented software engineering, we discussed the application of patterns for agent-based simulation models. We gave some examples that illustrated the particular situation in agent-based modeling showing that actual patterns in agent-based simulation can be different from the ones suggested for standard agent-based software.

One may observe that all tackled patterns in this contribution are on a very technical level. Yet, the actual modeling problem often starts before the technical level. Nevertheless, a list of well-defined, evaluated patterns on the level we discussed here, may show the technical options that a modeler has for realizing a model concept, especially when the pattern are connected to a particular modeling and simulation platform that includes some facility for code generation.

References

1. Willemain, T.: Insights on modeling from a dozen experts. Operations Research 42(2), 213–222 (1994)
2. Gamma, E., Helm, R., Vlissides, R.J.J.: Design Patterns: Elements of reusable object-oriented software. Addison Wesley, Boston (1995)

3. Triebig, C., Klügl, F.: Designing components for multiagent simulation. In: Agent-Based Modeling & Simulation Symposium at EMCSR, April 2006. Wien (2006)
4. Weiss, M.: Pattern-driven design of agent systems: Approach and case study. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 711–723. Springer, Heidelberg (2003)
5. Oluyomi, A., Karunasekera, S., Sterling, L.: A comprehensive view of agent-oriented patterns. Autonomous Agents and Multi-Agent Systems 15(3), 337–377 (2007)
6. Oluyomi, A., Karunasekera, S., Sterling, L.: Description templates for agent-oriented patterns. Journal of Systems and Software 81(1), 20–36 (2008)
7. Sauvage, S.: Agent oriented design patterns: A case study. In: AAMAS 2004: Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, Washington, DC, USA, pp. 1496–1497. IEEE Computer Society, Los Alamitos (2005)
8. Kendall, E.A., Krishna, P.V.M., Pathak, C.V., Suresh, C.B.: Patterns of intelligent and mobile agents. In: AGENTS 1998: Proc. of the 2nd Int. Conf. on Autonomous agents, pp. 92–99. ACM Press, New York (1998)
9. Aridor, Y., Lange, D.B.: Agent design patterns: elements of agent application design. In: AGENTS 1998: Proc. of the 2nd Int. Conf. on Autonomous agents, pp. 108–115. ACM Press, New York (1998)
10. Chacon, D., McCormick, J., McGrath, S., Stoneking, C.: Rapid application development using agent itinerary patterns. Technical Report 01-01, Lochheed Martin Advanced Technology Laboratories (2000)
11. Do, T.T., Kolp, M., Pirotte, A.: Social patterns for designing multiagent systems. In: Proc. of the 15th Int. Conf. on Software Engineering & Knowledge Engineering (SEKE 2003), San Franciso, USA, July 2003, pp. 103–110 (2003)
12. Hayden, S.C., Carrick, C., Yang, Q.: Architectural design patterns for multiagent coordination. In: Proc. of the 3rd Int. Conf. on Autonomous Agents (1999)
13. Oluyomi, A.: Pattern and Protocols for Agent-Oriented Software Engineering. PhD thesis, Department of Computer Science and Software Engineering, University of Melbourne, Australia (2006)
14. Klügl, F.: A validation methodology for agent-based simulations. In: SAC Symposium, Advances in Computer Simulation Track, Ceara, BR (March 2008)
15. Grimm, V., Railsback, S.F.: Individual-Based Modeling and Ecology. Princeton University Press, Princeton (2005)
16. Haefner, J.W.: Modeling Biological Systems – Principles and Applications, 2nd edn. Springer, New York (2005)
17. Koenig, R., Bauriedel, C.: Modular system of simulation patterns for a spatial-processes laboratory. In: Proc. of the ICA Workshop on Geospatial Analysis and Modeling, Vienna (July 2006)
18. Schütze, M., Riegel, J.P., Zimmermann, G.: A pattern-based application generator for building simulation. In: Jazayeri, M. (ed.) ESEC 1997 and ESEC-FSE 1997. LNCS, vol. 1301. Springer, Heidelberg (1997)
19. Bandini, S., Manzoni, S., Vizzari, G.: Towards a methodology for situated cellular agent based crowd simulations. In: Dikenelli, O., Gleizes, M.-P., Ricci, A. (eds.) ESAW 2005. LNCS (LNAI), vol. 3963, pp. 203–220. Springer, Heidelberg (2006)
20. Epstein, J.M., Axtell, R.: Growing Artificial Societies. Social Science from the Bottom Up. Random House Uk Ltd (1996)
21. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual. Addison Wesley, Reading (1999)

22. Caire, G., et al.: Agent oriented analysis using Message/UML. In: Wooldridge, M.J., Weiß, G., Ciancarini, P. (eds.) AOSE 2001. LNCS, vol. 2222, pp. 119–135. Springer, Heidelberg (2002)
23. Prechelt, L., Unger, B., Tichy, W.: Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. IEEE Transactions on Software Engineering 28(6), 595–606 (2002)
24. Vokác, M.: Defect frequency and design patterns: an empirical study of industrial code. IEEE Transactions on Software Engineering 30(12), 904–917 (2004)

Multi Criteria Decision Methods for Coordinating Case-Based Agents

Beatriz López, Carles Pous, Pablo Gay, and Albert Pla

University of Girona,
Campus Montilivi, edifice P4, Girona, Spain
{beatriz.lopez,carles.pous}@udg.edu, {pgay,apla}@eia.udg.edu
http://exit.udg.edu

Abstract. There is an increasing interest on ensemble learning since it reduces the bias-variance problem of several classifiers. In this paper we approach an ensemble learning method in a multi-agent environment. Particularly, we use genetic algorithms to learnt weights in a boosting scenario where several case-based reasoning agents cooperate. In order to deal with the genetic algorithm results, we propose several multi-criteria decision making methods. We experimentally test the methods proposed in a breast cancer diagnosis database.

Keywords: Ensemble Learning, Case-Based Reasoning, Multi Criteria Decision Making.

1 Introduction

Ensemble learning has been a matter of concern in the last recent years because of its benefits on reducing the bias-variance of classifiers. Bagging, boosting and staging are three very well known ways of addressing this relatively new way of learning. Bagging assigns randomly to each learner a set of examples, so the construction of complementary learners is left to the chance and to the instability of the learning methods. Boosting actively seeks to generate complementary base learners, on the basis of the methods of the previous learners. Staking deals with the combination of models of different algorithms [19].

Ensemble learning has been recently applied to multi-agent systems, so that several learning agents collaborate in a distributed environment. For example, in [13] the authors propose several ensemble schemas for cooperative case-based learners.

The usual way in which bagging and boosting integrate the different learners is under a weighted voting schema. Therefore, the key issue is the weight assigned to each agent. AdaBoost [4] is one of the best known learning algorithms for this purpose, having today a lot of variants. More recently, genetic algorithms (GAs) have also been applied [7], but mainly in non-multi-agent environments. Our research is related to extend the application of genetic algorithms, for boosting purposes, in a multi-agent environment where each classifier is linked to a given agent.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 54–65, 2009.

This multi-agent approach based on boosting can be applied in domains where there are different experts distributed for solving a problem, and each of them can provide a solution to the problem. In this scenario, each agent could kept in private the process of arriving at a given solution; the weight of each agent represents the confidence or trust of the agent when solving a problem. For example, in the medical domain, different units of a hospital can be involved in the diagnosis of a patient. Each unit has the corresponding physician responsible of the diagnosis according to his/her speciality: radiology, endocrinology, etc. Then, according to the data gathered in each unit, each physician (agent) can provide his/her diagnosis.

In particular, in our approach each classifier follows case-based reasoning (CBR) method, so we are dealing with CBR agents. Moreover, since we are actively seeking for the complementary of learners through a GA, we say that we are boosting CBR agents.

According to the genetic algorithm theory, several runs are required in order to deal with the randomness involved in this kind of algorithms [11]. Thus, two runs with different random-number seeds will generally produce different detailed behaviours. But finally, a single weight should be assigned to a boosting agent. In this paper, we present several alternatives for obtaining this single weight from the outcomes of the different genetic algorithm runs. Our methods have been applied in a breast cancer diagnosis domain, and we show the different results obtained. This paper extends the work in [8].

This paper is organised as follows. First we introduce the boosting schema in which our CBR agents cooperate. Next, we describe the GA we propose and the methods to manage the outcomes of the different runs. We continue by providing the information about the application domain we are working on and the results obtained in it. Finally, some related work is highlighted and some conclusion and discussion is provided.

2 Boosting CBR Agents

Our multi-agent system (MAS) consists of n case-based agents that cooperate for solving a problem. Each agent provides its advise or solution about a case, and a coordinator makes a final decision based on a weighted voted schema.

Each agent is trained with the same set of examples; however, each agent receives only a part of the examples (as in [12]). Thus, each agent is specialised in a particular field of knowledge of the domain.

Since there is a coordinator agent in charge of dealing with cooperation issues, the system is centralised. The coordinator agent keeps a $weight_i$ on each agent according to the performance provided by the agent. This weights are learned according to the method proposed in this paper in section 3.

When a new case C needs a solution, the coordinator agent broadcasts the case to the CBR agents. CBR agents compute internally a solution for the case following a case-based reasoning process (see [15] for further details). Next, the CBR agents reply to the coordinator with a tuple containing the class to which

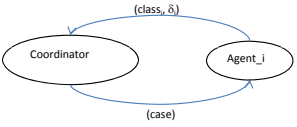


Fig. 1. Agent's interaction

the case belongs according to its case-base, and the confident it is on that solution (see Figure 1). That is:

$$a_i =< class_i, \delta_i > \tag{1}$$

where a_i is the answer of the agent i; $class_i$ the class provided by the agent; and δ_i is a confidence value in [0,1], where 1 means high confident. We are currently considering a diagnosis environment, so only two class values are under evaluation: 1 (positive diagnosis or illness) and 0 (negative diagnosis or healthy).

Afterwards, the coordinator agent combines the different answers in order to find information regarding the positive diagnostic according to the following expression:

$$v = \frac{\sum_{i=1}^{n} class_i * \omega_i}{\omega_i} \tag{2}$$

where n is the number of agents; and ω_i is a combination of the weight of the agent i and δ_i, such that $\omega_i = f(weight_i, \delta_i)$. The f function can be any, as for example the multiplication.

Then, if v is over a given threshold τ, the final answer of the multi-agent system is positive. Otherwise, negative. This decision procedure follows the reuse step of a case-based system as explained in [15]. See also [9] for further details on the boosting CBR MAS system.

3 Multi-Criteria Decision Making Methods for Coordination

The method we propose to learn agents' weights has two phases: genetic algorithm learning and multi-criteria decision processing.

3.1 Genetic Algorithm

A genetic algorithm (AG) consists on the following steps [11]:

1. Start with a randomly generated population of chromosomes
2. Calculate the fitness of each chromosome in the population
3. Repeat
   (a) Select a pair of chromosomes from the current population
   (b) With a probability p_c cross over the pair to form two offsprings
   (c) With a probability p_m mutate the two offsprings
4. Replace the current population with the new one
5. Goto step 2 until χ iterations.

As it is possible to observe, randomness plays a large role in each run; so two different runs can produce different results. Thus, averaged results on several runs should be obtained.

When applying genetic algorithms to learn the weights in a boosting CBR agents scenario, the key issues are how to represent chromosomes and how to define the fitness function. Particularly, we have defined the chromosome as an array of n values; each value represents the weight of an agent. On the other hand, the fitness of a chromosome is a function of the error of the boosting CBR system it codifies when applied to a data set of examples. So the chromosome is translated to the corresponding boosting CBR MAS, it is run for a given set of examples, and an averaged error over all of the examples is provided (see [9] for the details on the error computation). Finally, if there are no improvements between a population and the new one, 50 additional iterations are performed, and the GA is stopped. Regarding other details of the GA see also [9].

Given a set of M examples, m data sets can be generated according to a cross-validation methodology. Then, the GA is repeated for each set, obtaining m sets of weights together with the m error rates, one per each of the GA run.

3.2 Multi-Criteria Decision Methods

Multi-criteria decision making (MCDM) aims at supporting decisions when several alternatives are available according to different criteria [18]. We can order those alternatives, and then choose one. We can also combine all of them to obtain a new solution thanks to either information fusion techniques or aggregation operators. We are interested in the second option. Among the different aggregation operators, there are the mean (M) and the weighted mean (WM).

Thus, after m runs of the GA on boosting a number of n CBR agents, we get the following sets of weights:

run	Agent1	Agent2	...	Agentn
1	$weight_1^1$	$weight_2^1$	...	$weight_n^1$
2	$weight_1^2$	$weight_2^2$	...	$weight_n^2$
...	...	...	...	...
m	$weight_1^m$	$weight_2^m$	...	$weight_n^m$

The different runs can be considered alternatives from the MCDM point of view. Thus, a mean that can compute a final weight $weight_i$ for the i agent is the following:

$$weight_i = \frac{\sum_{j=1}^{m} weight_i^j}{m} \tag{3}$$

where m is the total number of weights obtained by the AG regarding the agent i.

In the case of a WM, we need to compute the mean values of the different weights $weigth_i^j$ obtained according to another ponderation $\mu_1, ...\mu_n$. Thus,

$$weight_i = \frac{\sum_{j=1}^{m} weight_i^j * \mu_i^j}{\sum_{j=1}^{m} \mu_i^j} \tag{4}$$

Table 1. *Left*: Example of weights obtained for n agents after m GA runs. *Right*: Rankings of the example.

run	A0	A1	A2	A3	A4	A5	A6	A7
1	0.74	0.06	0.05	0.01	0.04	0.00	0.07	0.03
2	0.63	0.03	0.01	0.00	0.27	0.00	0.04	0.01
3	0.42	0.01	0.06	0.04	0.35	0.02	0.08	0.03
4	0.79	0.01	0.04	0.01	0.08	0.02	0.02	0.02
5	0.02	0.00	0.01	0.09	0.79	0.05	0.02	0.01
6	0.83	0.02	0.03	0.01	0.03	0.03	0.03	0.01
7	0.90	0.02	0.01	0.01	0.01	0.01	0.04	0.01
8	0.35	0.04	0.03	0.04	0.42	0.04	0.06	0.03
9	0.24	0.02	0.02	0.03	0.64	0.01	0.04	0.01
10	0.85	0.01	0.04	0.03	0.03	0.01	0.02	0.01

run	A0	A1	A2	A3	A4	A5	A6	A7
1	1	3	4	7	5	8	2	6
2	1	4	6	8	2	7	3	5
3	1	8	4	5	2	7	3	5
4	1	7	3	8	2	5	6	4
5	5	8	7	2	1	3	4	6
6	1	6	4	7	2	3	5	8
7	1	3	6	5	8	4	2	7
8	2	6	8	5	1	4	3	7
9	2	6	5	4	1	8	3	7
10	1	6	2	4	3	7	5	8

So, a new parameter μ_i^j should be determined in order to obtain the final values $weight_i$. We propose four methods: rated ranking, voted ranking, error based, and mean value.

Rated ranking method. The rated ranking method consist on 1) ranking the different weights for a given set, and 2) computing the distance to the first position.

We can compute the ranking of each agent in all the sets by sorting them according to a descending order (from the highest to the lowest weight). So, we obtain a set of rankings as follows:

run	Agent1	Agent2	...	Agentn
1	$rank_1^1$	$rank_2^1$	...	$rank_n^1$
2	$rank_1^2$	$rank_2^2$	...	$rank_n^2$
...	...	...	...	...
m	$rank_1^m$	$rank_2^m$	...	$rank_n^m$

Finally, the μ_i^j is computed as follows:

$$\mu_i^j = \frac{1}{ranking_i^j} \tag{5}$$

In order to illustrate with an example this method, suppose that we have 8 agents (n=8) and we have run 10 times the GA (m=10). The weights obtained by the GA are shown in Table 1 left. Then, after sorting the values of the table, we get the rankings shown in Table 1 right. So agent $A0$ has been the agent with the highest weight in runs 1-4,6-7 and 10; while it occupies the second position in runs 8 and 9, and the fifth position in the 5th run.

Afterwards, the μ_i^j is computed according to equation 5. Finally, the weight of each agent, $weight_i$, is computed according to equation 4, obtaining the results shown in the "ranking" row of Table 3.

Voted ranking method. In this method, all the weights obtained by the GA $weight_i^j$ are ranked as in the previous one. However, after obtaining the ranking, we count the times an agent occupies the same position, obtaining the "voted" ranking for each agent vot_i^k. Thus if we have n agents, we have n possible votes per agent. In the next step, all the votes are averaged according to the following expression:

$$\mu_i^j = \frac{\sum_{k=1}^{n}[(n+1)-k] * \frac{vot_i^k}{n}}{n} \forall j \tag{6}$$

Observe, that μ_i^j as it is, can also be used as the $weight_i$, and we analyse this consideration in the results section.

Following the example of Table 1, we obtain the votes and μ_i^j values shown in Table 2. These values are then combined according to equation 4, and the agent weights are obtained are shown in the "voting" row of Table 3.

Error based method. In this method we want to take advantage of the information provided by the learning algorithm related to the error $error^j$ to which the GA (run j) converges. Thus, the distance to the error is used as the μ_i^j, as follows

$$\mu_i^j = 1 - error^j \tag{7}$$

In our example, we got the following error rates for every GA run: 0.29, 0.30, 0.27, 0.31, 0.29, 0.33, 0.38, 0.26, 0.26, 0.34. Therefore, the weights obtained for our agents with this method are the ones shown in the "error" row of Table 3.

Mean value method. Now, we define a method based on the mean weight value obtained for the agents in all the runs. Let mv_i be this mean value. Then, the inverse to the distance to this value is used as μ_i^j. That is,

$$\mu_i^j = 1 - |mv_i - weight_i^j| \tag{8}$$

In our example, we get the following mv^j values (from Table 1): 0.58, 0.02, 0.03, 0.03, 0.27, 0.02, 0.04, 0.02. Finally, the weights obtained for each agent as shown in the "mean value" row of Table 3.

Table 2. Votes and μ_i^j values from the example of Table 1

Rank	A0	A1	A2	A3	A4	A5	A6	A7
1	7	0	0	0	3	0	0	0
2	2	0	1	1	4	0	2	0
3	0	2	1	0	1	2	4	0
4	0	1	3	2	0	2	1	1
5	1	0	1	3	1	1	2	1
6	0	4	2	0	0	0	1	3
7	0	1	1	2	0	3	0	3
8	0	2	1	2	1	2	0	2
μ_i^j	0.93	0.41	0.51	0.44	0.79	0.43	0.68	0.33

4 Application to Breast Cancer Diagnosis

In this section we provide a brief description of the data base used for experimentation. Afterwards, some details about the experimental setup are presented. Next, the results obtained are shown.

4.1 Experimental Set Up

We have used a Breast Cancer data base provided by the team of physicians we are working with. It consists of 871 cases, with 628 corresponding to healthy people and 243 to women with breast cancer. There are 1199 attributes for each case. Since there are redundant, wrong and useless information a preprocess was carried out. Also, the preprocess was used to obtain data corresponding to independent individuals, since there are patients in the database that are relatives. As consequence, the database was constituted of 612 independent cases, with 239 healthy people.

A first selection of the relevant attributes was performed by the physicians. According to their knowledge, 85 attributes were selected, being 46 of them numerical and the remaining categorical.

Data has been partitioned in 8 groups, following the questionnaire criteria with which physicians have collected them that are related to different medical specialisations (epidemiology data, family information data, etc.). Each group of data has been assigned to an agent. Therefore, we have 8 CBR agents in our system.

We have followed a cross-validation procedure, with 90% of the cases for training and 10% for testing. Up to 10 folds were generated, and 10 AG runs have been performed, one per fold. Thus, we finally obtain 10*8 weights. The eXiT*CBR tool [16] has been used since it allows to easily define CBR boosting agents in a parameter-based way.

The following experimental settings have been defined:

- None: no learning has been applied. So all the agents weights have been set to 1.
- Mean: the mean operator has been used
- Ranking: the WM has been used together with the rated ranking method
- Voting: the WM operator has been applied together with the voting method
- Error: the WM operator has been used together with the error-based method
- MeanValue: the WM operator has been used together with the mean value method.

The results obtained in each experimental configuration are detailed in the next section.

4.2 Results

The weights obtained are the ones shown in Table 3. Regarding the outputs of the boosting CBR MAS, we have used ROC curves. ROC (*Receiver Operator Characteristics*) curves depict the tradeoff between hit rate and false alarm rate [3].

Table 3. Weight obtained after applying the different MCDM methods

	A0	A1	A2	A3	A4	A5	A6	A7
None	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Mean	0.58	0.02	0.03	0.03	0.27	0.02	0.04	0.02
Ranking	0.55	0.01	0.01	0.01	0.22	0.01	0.01	0.00
Voting	0.54	0.01	0.02	0.01	0.21	0.01	0.03	0.01
Error	0.39	0.02	0.02	0.02	0.19	0.01	0.03	0.01
MeanValue	0.45	0.02	0.03	0.03	0.19	0.02	0.04	0.02

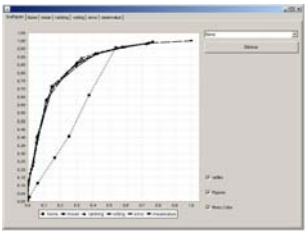


Fig. 2. Comparison of the different scenarios

Figure 2 shows the plot corresponding to the different scenarios. As it is possible to observe, the worst situation is when all of the agents weights are set to 1 (so the boosting voting schema has no weight). The remaining methods perform quite well and in a similar behavior. Analysing in detail the results, we obtain the following AUC (Area Under the Curve) values: None 0.706, Mean 0.851, Ranking 0.854, Voting 0.852, Error 0.853, MeanValue 0.848. The AUC values are quite similar too, being the ranking method the one that outperforms the others. Analysing the weights obtained by all of our methods, we see that in fact, the weights are quite closer. So we are not surprised on obtaining so close results. However, this does not happen with the weights obtained in each AG run, as shown in Table 1.

We have also analysed the results of our methods compared to a single AG run. The different situations are being analysed are the following:

- run1: the weights of the first GA run have been set in the boosting CBR MAS, and run over all the folders used for all the GA.
- run1_newfolder:the weights of the first GA run have been set in the boosting CBR MAS, and run the MAS over a new set of folders.
- ranking: the weights according to our ranking method, and run using the same folders than the GA (the same than in the run1).
- ranking_newfolder: the weights obtained with our ranking method, and run the MAS over the new set of folders (the same than in the run1_newfolder)

Figure 3 right shows the results. We can see that the performance of the learnt weights are similar. However, in Figure 3 left, we have changed the weights

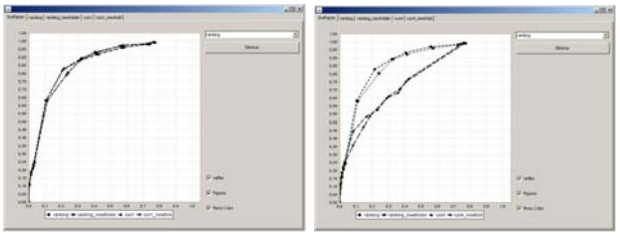


Fig. 3. Comparison of boosting CBR performance when examples change

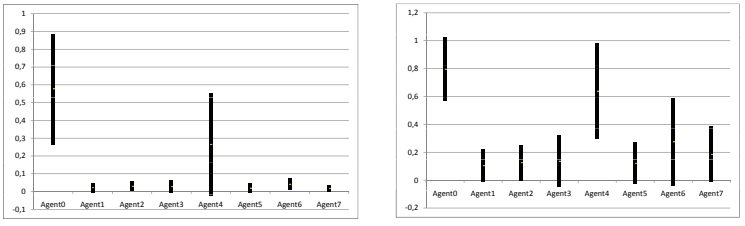


Fig. 4. Mean and standard deviation a) *top*: after 10 runs; b) *bottom*: after 50 runs

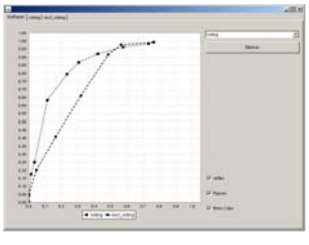


Fig. 5. Consideration of a voting based exclusively on the ranking

obtained in the first run by the ones obtained in the fourth run. Both runs stop with a close error value: the first run at 0.29, and the fourth run at the error value of 0.31, quite closer to the previous one. So these results confirm the need of running the AG several times and finding an aggregated value. The aggregated results maintain the behaviour.

We have repeated the experimentation increasing the number of folds up to 50. Results, however, were not so good. The AUC obtained are the following: None 0.706, Mean 0.833, Ranking 0.827, MeanValue 0.848. The reason for that is that the variance on the weights obtained by the AG is higher. Figure 4 left shows the mean an standard deviation of the weights obtained with the 10 folds, while in Figure 4 right the ones obtained for the 50 folds. As it is possible to observe, the variance with 50 folder has increased. So, in a future work we need to determine the adequate number of runs required in order to obtain the appropriate results.

Finally, we have also used the value of μ_i^j as the weights $weight_i$, since it is the same for all the runs. That means, that we do not weight the AG results, but we are using the qualitative information about the ranking of the agents on the results. Results obtained are shown in Figure 5, in which the excl_voting line is the one corresponding to this experiment, and compared to the previous voting results. We can see that the voting schema, as we have proposed in section 3, is the best.

5 Related Work

There are several works related to boosting CBR agents in particular, and ensemble learning in general [17,10]. For example, in [13] two schemas are proposed: bagging and boosting. In this work, the authors focus on how cases should be shared among agents. We are not so worried about that, but in how to set up the weights assigned to the agents thanks to a GA methodology.

A work closer to us is [12], in which the authors propose a corporate memory for agents, so that each agent knows about a piece of the case, as in our case. In [12], however, the authors propose a negotiated retrieval method based on distributed constraint optimisation techniques. We are using the basic weighting voting schema for combining CBR agents.

SEASALT [1] presents also a multi-agent architecture of case-based reasoning agents. The application involved in this research, however, is related to compound a solution from the sequence of different solutions provided by CBR agents, instead of adopting a single solutions by a social choice method as presented in this paper. Nevertheless, we need to study in deeper this architecture in order to incorporate some of the proposals regarding knowledge organisation.

Regarding research works on the use of GA in a boosting environment, it is important to distinguish the approach followed in [7]. Here, the authors analyse the greedy behaviour of Adaboost and suggest to use GAs to improve the results. Another interesting work is [5], in which the AGs are used to search on the boosting space for sub-ensembles of learners.

There are also some approaches that boost GAs. For example, in [14] a two stage method is proposed, in which in a first step, genetic fuzzy classifiers provide sets of rules, and in a second boosting stage, these classifiers are combined. Nevertheless, our goal is the opposite to these works: we are not boosting GAs but using GA to combine classifiers.

Finally, some interesting studies to which our research work is related is [6]. In this case, agent's trust is studied under the evolutionary paradigm. We believe that our approach based on specialised agents is equivalent to it. This view of ensemble weights as trust has also been studied in [2].

6 Conclusions

Boosting mechanism are a promising paradigm for multi-agent systems. In this paper we have described a boosting mechanism based on CBR agents, in which

the final result of the system is the weighted voting results of the different agents. In order to determine the weights, we are using genetic algorithms. Due to the randomness involved in GA, it is necessary to run several times the GAs, obtaining different results. In this paper we present and analyse different multicriteria decision making methods in order to deal with the different GA results, that allow to determine a single weight for each agent.

The methods have been applied to a breast cancer diagnosis data base. The results show that MCDM methods obtain weights that are more robust to changes on the examples. Among all the methods presented, the one based on the ranking of the agents in each AG is the one that outperforms the other, although the results are quite close.

Acknowledgments. This research project has been partially funded by the Spanish MEC projects DPI 2006-09370, CTQ2008-06865-C02-02/PPQ, TIN2008-04547/TIN, and Girona Biomedical Research Institute (IdiBGi) project GRCT41.

References

1. Bach, K., Reichle, M., Reichle-Schmehl, A., Althoff, K.-D.: Implementing a coordination agent for modularised case bases. In: AI 2008, 13th UK Workshop on CCBR, pp. 1–12 (2008)
2. Birk, A.: Boosting cooperation by evolving trust. Applied Artificial Intelligence 14, 769–784 (2000)
3. Fawcett, T.: Roc graphs: Notes and practical considerations for data mining researchers. Technical Report HPL-2003-4., HP Labs (2003)
4. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boostingg. Journal of Computer and System Sciences 55(1), 119–139 (1997)
5. Hernández-Lobato, D., Hernández-Lobato, J.M., Ruiz-Torrubiano, R., Valle, A.: Pruning adaptive boosting ensembles by means of a genetic algorithm. In: Corchado, E., Yin, H., Botti, V., Fyfe, C. (eds.) IDEAL 2006. LNCS, vol. 4224, pp. 322–329. Springer, Heidelberg (2006)
6. Komathyk, K., Narayanasamy, P.: Trust-based evolutionary game model assisting aodv routing againsts selfishness. Journal of network and computer-application 31(4), 446–471 (2008)
7. Yalabik, Ï., Yarman-Vural, F.T., Uçoluk, G., Sehitoglu, O.T.: A pattern classification approach for boosting with genetic algorithms. In: 22th International Symposium on Computer and Information Sciences, pp. 1–6 (2007)
8. López, B., Pous, C., Gay, P., Pla, A.: Multi criteria decision methods for boosting CBR agents with genetic algorithms. In: AAMAS workshop on Adaptative and Learning Agents, pp. 55–209, 58 (2009)
9. López, B., Pous, C., Pla, A., Gay, P.: Boosting CBR agents with genetic algorithms. In: McGinty, L., Wilson, D.C. (eds.) ICCBR 2009. LNCS (LNAI), vol. 5650, pp. 195–209. Springer, Heidelberg (2009)
10. Martin, F.J., Plaza, E., Arcos, J.L.: Knowledge and experience reuse through communication among competent (peer) agents. International Journal of Software Engineering and Knowledge Engineering 9(3), 319–341 (1999)

11. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, Cambridge (1998)
12. Nagendra-Prasad, M.V., Plaza, E.: Corporate memories as distributed case libraries. In: 10th Banff Knowledge Acquisition for Knowledge-based Systems Workshop, pp. 1–19 (1996)
13. Ontañon, S., Plaza, E.: Cooperative multiagent learning. In: Alonso, E., Kudenko, D., Kazakov, D. (eds.) AAMAS 2000 and AAMAS 2002. LNCS (LNAI), vol. 2636, pp. 1–17. Springer, Heidelberg (2003)
14. Özyer, T., Alhajj, R., Barker, K.: A boosting genetic fuzzy classifier for intrusion detection using data mining techniques for rule pre-screening. In: Design and application of hybrid intelligent systems, pp. 983–992. IOS Press, Amsterdam (2006)
15. Pous, C., Gay, P., Pla, A., Brunet, J., Sanz, J., López, B.: Modeling reuse on case-base reasoning with application to breast cancer diagnosis. In: Dochev, D., Pistore, M., Traverso, P. (eds.) AIMSA 2008. LNCS (LNAI), vol. 5253, pp. 322–332. Springer, Heidelberg (2008)
16. Pous, C., Gay, P., Pla, A., López, B.: Collecting methods for medical CBR development and experimentation. In: Schaaf, M. (ed.) Workshop CBR in the Health Sciences (ECCBR-HC), pp. 89–98. Tharax-Verlag (2008)
17. Teodorescu, E.I., Petridis, M.: An architecture for multiple heterogeneous case-based reasoning employing agent technologies. In: CIMAS (2008), http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-375/
18. Torra, V., Narukawa, Y.: Modeling Decisions: Information Fusion and Aggregation Operators. Springer, Heidelberg (2007)
19. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)

Agent Cooperation for Monitoring and Diagnosing a MAP

Roberto Micalizio and Pietro Torasso

Dipartimento di Informatica, Università di Torino, Torino, Italy
{micalizio,torasso}@di.unito.it

Abstract. The paper addresses the tasks of monitoring and diagnosing the execution of a Multi-Agent Plan, taking into account a very challenging scenario where the degree of system observability may be so low that an agent may not have enough information for univocally determining the outcome of the actions it executes (i.e., pending outcomes).

The paper discusses how the ambiguous results of the monitoring step (i.e., trajectory-set) are refined by exploiting the exchange of local interpretations between agents, whose actions are bounded by causal dependencies. The refinement of the trajectory-set becomes an essential step to disambiguate pending outcomes and to explain action failures.

1 Introduction

The problem of supervising the execution of a multi-agent plan (MAP) is receiving an increasing attention; in fact, the idea of distributing the execution of a complex plan among a number of cooperating agents, which execute actions concurrently, has proved to be quite useful for several domains and applications. The supervision is a complex task as one has to take into account *plan threats*, which can cause action failures. In the last few years some approaches have been proposed to attack the problem ([1,2,3]). Typically these approaches assume that action failures are not consequences of plan flaws, but failures are due to the occurrence of exogenous events (such as unexpected changes in the environment, occurrence of faults in some agents functionalities, etc.). Moreover, because of causal dependencies between actions executed by different agents, the failure in an action assigned to an agent may impact also the execution of the actions assigned to other agents. For this reason, it is necessary to perform a plan diagnosis in order to detect an action failure as soon as possible, and to single out (if possible) the reason for such a failure. In fact, the ability of an agent to perform some form of the plan recovery and repair strongly depend on the capabilities of inferring a precise diagnosis (see for example [4]).

In this paper we advocate a distributed approach to plan supervision, where each agent is responsible for supervising (monitoring, detecting action failures and performing plan diagnosis) the actions it executes. In particular, action models represent not only the nominal action behavior, but also the (usually non deterministic) anomalous behavior due to the occurrence of exogenous events. Of course, the adoption of non deterministic action models make the supervision task even more complex.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 66–78, 2009.

Moreover, since the system is distributed, each agent has just a limited view of the progress of the global plan; in fact it receives only partial observations from the environment. As a consequence an agent cannot, in general, precisely detect the outcome (success or failure) of its actions on the sole basis of the observations it receives. The proposed solution involves the exchange of local interpretations of plan execution (i.e., action outcomes) among agents, in order to refine the local point of view of each agent, and possibly to detect and explain action failures.

The paper is organized as follows: first, we introduce the basic notions of global and local plans, then we formalize the processes of monitoring and diagnosis of a MAP and discuss the communication and cooperation among agents for inferring the action outcomes which cannot be immediately determined.

2 Distributed Execution of a Multi-Agent Plan

In this paper we consider a specific class of multi-agent systems which can be modelled as Multi-Agent Plan (MAP). In a MAP, a team $\mathcal{T}$ of agents strictly cooperate to reach a common, complex goal G by exchanging one another services and this cooperative behavior introduces causal (and precedence) dependencies among their activities.

Global plan. The MAP P is modeled as the tuple $\langle A, E, CL, CC, NC\rangle$ (see e.g., [5]) such that: A is the set of the action instances a the agents have to execute, each action a is assigned to a specific agent in the team; E is a set of precedence links between action instances, CL is a set of causal links of the form $l : a \xrightarrow{q} a'$; the link l states that the action a provides the action a' with the service q, where q is an atom occurring in the preconditions of a'. Finally, CC and NC are respectively the *concurrency* and *non-concurrency* symmetric relations over the action instances in A: a pair $\langle a, a'\rangle$ in CC models a joint action, while constraints in NC prevent conflicts for accessing the resources.

To keep the discussion simple, in this paper we do not consider joint actions even though the approach can be extended to deal with them (see e.g. [3]). Moreover, we translate the non-concurrency constraints into precedence links, so that during the plan execution agents do not need to negotiate for accessing resources (this is equivalent to the *concurrency requirement* introduced in [2]); in particular, each non concurrency constraint $\langle a, a'\rangle \in NC$ (ruling the mutual exclusion access to a resource res), is substituted either with $a \prec_{res} a'$ or with $a' \prec_{res} a$. The result of this translation procedure is a simple form of scheduling, where actions assigned to different agents are explicitly related by means of a (partial) precedence relation; formally, the MAP P to be executed (and supervised) is defined as $P{=}\langle A, E, CL, RE\rangle$, where RE is the set of precedence links ruling the access to the resources.

Local Plans. Since the MAP P is executed in a distributed way by the agents in the team, each single agent is responsible just for a portion of a MAP called *local plan*. Intuitively, a local plan P^i is the projection of the global plan P over the action instances assigned to the agent i. Thus, the plan P is decomposed into

as many local plans as the agents in $\mathcal{T}$; however, the decomposition must keep trace of the causal and precedence relations existing between action instances assigned to different agents: the local plan P^i for agent i is formally defined as the tuple $P^i{=}\langle A^i, E^i, CL^i, T^i_{in}, T^i_{out}, RE^i_{in}, RE^i_{out}\rangle$, where A^i, E^i and CL^i have the same meaning of the sets A, E and CL, respectively, restricted to actions assigned to agent i. The remaining sets are used to keep a trace of the dependencies existing between actions belonging to different local plans; in particular, the causal links from (to) an action of agent i to (from) an action of another agent j are collected in the sets T^i_{out} (outgoing links) and T^i_{in} (incoming links), respectively. Similarly, the precedence links for accessing the resources are subdivided into RE^i_{in} (incoming links) and RE^i_{out} (outgoing links).

In order to simplify the discussion, we assume that each local plan P^i is *totally ordered*, thereby it can be represented as the sequence of actions $\langle a^i_0, a^i_1, \ldots, a^i_\infty\rangle$.

Distributed execution and coordination under nominal conditions. As an effect of the decomposition of the global plan, the execution of the local plans is performed by the agents concurrently and asynchronously. In particular, we assume an agent executes its next action a as soon as the preconditions of a are satisfied; on the contrary, the agent will wait as long as the preconditions of a are not satisfied.

Because of the causal and precedence constrains introduced by the planning step, the agents have to coordinate one another by exchanging messages. All the knowledge required for inter-agent communication is encoded in P^i: every outgoing causal (or precedence) link in T^i_{out} (RE^i_{out}) is associated with a *send-message* operation toward agent j. Similarly, every incoming causal (or precedence) link in T^i_{in} (or RE^i_{in}), is associated with a *receive-message* operation[1].

Let us suppose that a causal link in T^j_{in} states that the preconditions for action a (to be executed by agent j) involve a service q provided by agent i: the agent j will wait a message from i about the service q before executing a.

Plan threats. The nominal execution of the local plan may be affected by *plan threats*, which typically cause action failures. In this paper, plan threats are exogenous events which cause abrupt changes in the agent status; we will denote as $\mathcal{E}$ the set of exogenous events which may occur during the plan execution; while we will use the symbol ϵ to represent the absence of exogenous event.

Observe that communication and cooperation among agents are not only needed under nominal execution of the plan, but are even more important in presence of some failure. In fact, when agent i realizes that agent j cannot be provided with service q due to an action failure, it has to exploit the causal link in T^i_{out} to send agent j a message about the non availability of service q: agent j becomes aware that the preconditions of action a will never be satisfied.

The cooperation and communication becomes more critical if we want to deal with cases when the system observability is so partial that an agent i is unable to conclude that the service q has been provided or not, and therefore it cannot guarantee the agent j that q has been achieved.

[1] We assume that the inter-agent communication is reliable and instantaneous.

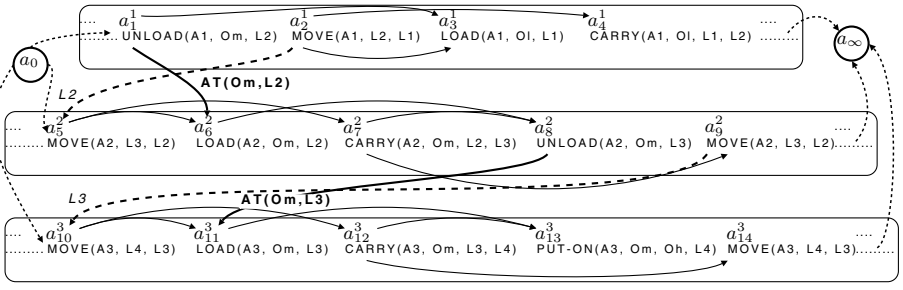


Fig. 1. The global plan to be monitored

In this way, agent j cannot be sure whether the preconditions of its next action a (which include q) are satisfied or not. If j adopted a conservative policy, it would not execute action a, and hence it would stop the execution of its local plan; however, this kind of policy may result impractical in many domains. We propose a weak commitment policy where we assume by default that service q has been provided, so the agent j can execute the action a. Obviously such a default assumption can be wrong in some cases and therefore the mechanisms for detecting the action failure and for performing plan diagnosis are much more complex when respect to the case no default assumption is made.

Running Example. Throughout the paper we will illustrate the proposed methodology by means a simple example from the blocks world. Let us consider three agents that cooperate to achieve a global goal G where a number of objects `O1`,..., `On` must be moved from a source location `L1` to a target position `L4`, passing through the intermediate positions `L2` and `L3`. All these positions are critical resources as only one agent can access one of them at a given time instant; moreover, these positions are the only locations where, under nominal conditions, an agent can pick up or release a block. Figure 1 shows a portion of a MAP achieving the goal G; in particular the picture shows the actions involved in the delivery of the object `Om`: each agent is responsible for carrying the object from a position to the next one: `A1` from `L1` to `L2`, `A2` from `L2` to `L3`, and `A3` from `L3` to `L4` where `Om` is put on the top of a stack of objects.

The three rectangles enclose the local plans assigned to the agents. It is easy to see that the MAP is a DAG where nodes are action instances and edges represent precedence or causal links. However, for the sake of readability, internal precedence links (i.e., between actions in the same local plan) are not displayed, while internal causal links (thin solid edges) are reported without labels. Instead, the picture highlights causal (solid edges) and precedence (dashed edges) links between actions in different local plans: these links represent relations between agents. For example, the causal link between actions a_1 and a_6 means that the agent `A1` provides agent `A2` with the service *at(Om, L2)* (i.e., the object `Om` is located in position `L2`); whereas the precedence link between actions a_2 and a_5, labeled with the resource id `L2` means that action a_5 can be executed only after

the execution of action a_2, i.e., only when the resource L2 is no longer used by agent A1 and it is made available to agent A2.

3 Basic Concepts on Distributed MAP Monitoring

In this section we introduce the model-based methodology we adopt for monitoring the execution of a MAP. In particular, the models we propose for the agent state and actions take care of the inherent ambiguity of the system.

Agent state. Intuitively, the status of the system can be represented in terms of the status variables of the agents in the team $\mathcal{T}$ and of the status of the global resources *RES* available in the environment. However, given the decentralized setting, the status of the system has to be represented in a distributed way by considering the set VAR^i of status variables associated to each agent i. As usual in approaches based on Discrete Event Systems (DESs), each variable $v \in VAR^i$ assumes values in a predefined and finite domain $Dom(v)$.

The set of status variables VAR^i is partitioned into two subsets: END^i and ENV^i. END^i includes the endogenous variables which characterize the specific agent i (and therefore there is no direct relation between the endogenous variables of two agents i and j); ENV^i includes all the variables concerning the environment (e.g., the status of a resource, or the position of an object). Note that, because of the partitioning, each agent i maintains a private copy of the variables in ENV^i; more precisely, for each resource $res_k \in RES$ $(k : 1..|RES|)$ the private variable $res_{k,i}$ is included in the set ENV^i. The consistency among the private copies is guaranteed by the existence of precedence links in RE: the status of a resource is known only by the agent that holds it, for all the other agents the status of the resource is *not-available*.

As noted earlier, the relinquishment/acquisition of a resource happens through the exchange of messages between the agent who releases the resource and the agent who gets the resource (this is performed via the send and receive actions associated to the precedence link).

Action models. The model of an action a takes into account not only the nominal effects of the action, but also the non deterministic effects of exogenous events affecting the action execution. Formally, an action model is the tuple $\langle pre(a), eff(a), event(a), \Delta(a)\rangle$ where $pre(a)$ and $eff(a)$ are subsets of VAR^i, representing the variables over which the preconditions and the effects are defined, respectively; $event(a)$ is a subset of exogenous events in $\mathcal{E}\cup\{\epsilon\}$ that may occur during the execution of a. Finally, $\Delta(a)$ is a transition relation modeling how the status of agent i changes after the execution of a. In particular, the state transition $\langle s_l, \epsilon, s_{t+1}\rangle \in \Delta(a)$ models the case of the nominal behavior: ϵ denotes the case where there is no occurrence of any exogenous event, while s_l and s_{l+1} represent two agent states (i.e., two complete assignments of values to the status variables in VAR^i) at the steps l and $l+1$, respectively. The state transition $\langle s_l, e, s_{l+1}\rangle$ (with $e \in \mathcal{E}$), denotes the occurrence of the exogenous event e; note that it is easy to model, within the transition relation $\Delta(a)$, the non deterministic effects of e. Since in some domains it could be impossible (or too costly)

to provide a complete model of the effects of a exogenous event, we extend the domain $Dom(v)$ of each variable $v \in VAR^i$ by including the value *unknown*; when a variable assume the value *unknown* in s_{l+1} the actual value of v is no more predictable.

Running example. Let us assume that, in our blocks world example, the agent `A2` loses the object `Om` while `A2` is moving from `L2` to `L3` (action a_7 of Fig. 1). To take into account this possibility, the transition relation of the *carry* action includes, among others, a state transition describing the effects of the exogenous event *lose-object* on the status of agent `A2`. Intuitively, the *lose-object* event changes the value of the variable `A2`.*carrying* from `Om` to *empty*; at the same time, the variable `Om`.*position* (representing the position of the object) changes from *on-board*-`A2` to *unknown*; in this case, in fact, it is unrealistic to have a precise model for the *lose-object*, and some of its effects cannot be anticipated.

4 Dealing with Ambiguity in MAP Monitoring

In the distributed framework proposed in this paper, each agent is responsible for monitoring the actions it executes. Intuitively, the monitoring task has to *keep track* of the agent status while the agent is executing the actions in its local plan, and to*detect* as soon as possible anomalous discrepancies between the expected nominal behavior of the agent and the observed one. To meet the first objective, each agent i maintains a "history", namely a *trajectory*, representing the sequence of state transitions occurred during the execution of a plan segment. To meet the second objective, the agent i must be able to determine the outcome of the actions it has executed; in fact, an anomalous execution manifests itself when the nominal effects of an action have not been achieved.

The monitoring task is made complex not only because of the non determinism of the action model, but also because of the very partial observability of the system, which impacts the monitoring process in two ways: first, the trajectory of agent i cannot be precisely estimated, (and therefore a set of alternatives, called *trajectory-set*, must be maintained); second, the agent i must be able to deal with ambiguous action outcomes: i could not be able to determine whether the nominal effects of an action have been achieved.

Agent Trajectory and Trajectory-set. An *agent trajectory*, denoted as $tr^i(0,l)$, is defined over a segment $[a_0^i,\ldots,a_l^i]$ of the local plan P^i, and consists of an ordered sequence of agent states and exogenous events representing an evolution of the status of agent i consistent with the observations it has received so far, more formally:

Definition 1. *The agent trajectory* $tr^i(0,l)$ *over the plan segment* $P^i[a_0,\ldots,a_l]$ *is* $tr^i(0,l)=\langle s_0,e_0,s_1,\ldots,e_l,s_{l+1}\rangle$, *where:*

s_k $(k:0..l+1)$ *is the state of agent* i *at the* k*-th step such that* $obs^i(k)\cup s_k \not\vdash \bot$.

e_h $(h:0..l)$ *is an event in* $\mathcal{E}\cup\{\epsilon\}$ *occurring during execution of action* a_h, *involved in the agent state transition from* s_h *to* s_{h+1}.

As mentioned above, we do not assume that the available system observability guarantees to precisely determine the status of an agent after the execution of each action. As a consequence of this ambiguity, the structure that the agent i has to maintain is the *trajectory-set* $Tr^i[0..l]$, which includes all the possible agent trajectories $tr^i(0,l)$ consistent with the observations received during the execution of the plan segment $P^i[a_0,\ldots,a_l]$.

Albeit the trajectory-set $Tr^i[0,l]$ maintains the history in the interval $[0,l+1]$, it is sometimes useful to single out the agent *belief state* at a given step k :

Definition 2. *Given the consistent trajectory-set* $Tr^i[0..l]$*, the agent* belief state $\mathcal{B}^i_k$ *(*$k : 0..l+1$*) is the set of all the consistent agent states inferred at the* k*-th step. Formally,* $\mathcal{B}^i_k = \text{PROJECTION}_{s_k}(Tr^i[0..l])$.

Action Outcomes. Intuitively, the outcome of an action a is a synthetic piece of information which states whether the nominal effects of a have been achieved or not; of course, in the positive case the outcome of a is *succeeded*, the outcome of a is *failed* otherwise.

In the Relational framework we propose, the nominal effects of an action a result from the following expression:

$$\mathit{nominalEff}(a) = \text{PROJECTION}_{\mathit{eff}(a)}(\text{SELECT}_{e=\epsilon}\Delta(a)).$$

Namely, $\mathit{nominalEff}(a)$ is the complete assignment of values to the status variables in $\mathit{eff}(a)$, when no exogenous event occurs (i.e. $e = \epsilon$).

The main problem in assessing the outcome of an action is the inherent ambiguity both in the action model and in the trajectory-set. For instance, in order to assess the outcome action a^i_l, agent i needs to check whether the nominal effects of a^i_l are satisfied in the agent belief state $\mathcal{B}^i_{l+1}$. Unfortunately, given the partial system observability, the agent belief state $\mathcal{B}^i_{l+1}$ is in general ambiguous: it contains states where the nominal effects of a^i_l hold, and other states where they do not hold. The following, conservative definitions allows agent i to univocally determine the success or the failure of action a^i_l.

Definition 3. *The outcome of action* a^i_l *is*

succeeded *iff for each state* $s \in \mathcal{B}^i_{l+1}$, $s \vdash \mathit{nominalEff}(a^i_l)$.
failed *iff for each state* $s \in \mathcal{B}^i_{l+1}$, $s \cup \mathit{nominalEff}(a^i_l) \vdash \bot$

However, in all those cases where these two definitions are not applicable we adopt a weak commitment policy which allows the outcome of an action to be *pending*; the assessment of the outcome is postponed till the belief state $\mathcal{B}^i_{l+1}$ is sufficiently refined to conclude either the success or the failure. To this end, the agent i maintains a list pO^i of actions whose outcome has not been (yet) determined.

The incremental monitoring process. Let us assume that the action a^i_l requires as precondition the service q provided by action a^j_m (i.e., by another agent j). Thus, as soon as action a^j_m has been executed, agent j has to notify agent i about the result of such an action by sending a message $msg(a^j_m, a^i_l)$ to agent i. Such a message includes only the tuple $\langle \epsilon, q \rangle$ in case agent i can univocally

detect the success of action a_m^j (i.e., no exogenous event occurred and the service q has been provided). In case the outcome of action a_m^j is pending (i.e., agent j is not sure about the achievement of service q), the message includes not only the nominal situation $\langle \epsilon, q \rangle$, but also the anomalous situation $\langle exo(j, a_m^j), \neg q \rangle$, where the service q is not achieved because of the occurrence of an exogenous event denoted as $exo(j, a_m^j)$.

Since the action a_m^j should provide the service q to action a_l^i, the agent i has to consume the message $msg(a_m^j, a_l^i)$ during the monitoring of action a_l^i; in particular:

Definition 4. $Tr^i[0,l] = \text{SELECT}_{obs^i(l)}[[Tr^i[0,l-1] \times msg(a_m^j, a_l^i)] \text{ JOIN } \Delta(a_l^i))]$

Intuitively, the Relational product $Tr^i[0,l-1] \times msg_l^i$ is the mechanism through which the agent i includes within its trajectory-set the info provided by agent j. The join operation appends each possible transition in $\Delta(a_l^i)$ at the end of each trajectory $tr^i(0,l-1)$ in $Tr^i[0,l-1]$ iff the agent state s_l (i.e., the last agent state in the agent trajectory included the info provided by the message) satisfies the preconditions of the action a_l^i. The selection operation prunes the estimations by removing all the predicted trajectories which are inconsistent with the observations received as a feedback for the execution of a_l^i. It is important to note that the selection operator has an impact on the whole trajectory-set: the agent trajectory $tr^i(0,l)$ is removed from $Tr^i[0,l]$ when the last agent state s_{l+1} is inconsistent with the observations. As we will discuss later, through this mechanism it is possible to refine the past history of the agent status, and possibly determine the outcome of some past actions.

However, the above mechanism is not complete: it is possible, in fact, that for some trajectories $tr^i(0,l-1) \in Tr^i[0,l-1]$, the last state s_l does not match any of the transitions in $\Delta(a_l^i)$; this happens when some variables in s_l mention the symbol *unknown* (i.e., that variables are no longer predictable). In order to be able to incrementally extend also these trajectories for which the model of the action is not directly applicable, the monitoring step makes use of a weak prediction model:

Definition 5. *Let $tr^i(0,l-1)$ be an agent trajectory in $Tr^i[0,l-1]$ such that s_l does not satisfy the preconditions for action a_l^i; the agent trajectory $tr^i(0,l-1)$ is extended by appending a new state transition $\langle s_l, \epsilon, s_{l+1} \rangle$, where for each variable $v \in VAR^i$:*

v assumes the value unknown *in s_{l+1} iff $v \in \text{eff}(a_l^i)$*

v assumes in s_{l+1} the same value assigned in s_l otherwise

The first condition states that the effects of the action a_l^i become no longer predictable; the second condition imposes the persistency for all those variables which are not included in the definition of the nominal effects of the action a_l^i.

Observe that also the predictions inferred by means of the weak model must be consistent with the observations; when a variable v is *unknown* in s_l, it assumes in s_{l+1} the observed value in $obs(a_l^i)$, that is, the value *unknown* matches with any possible observation.

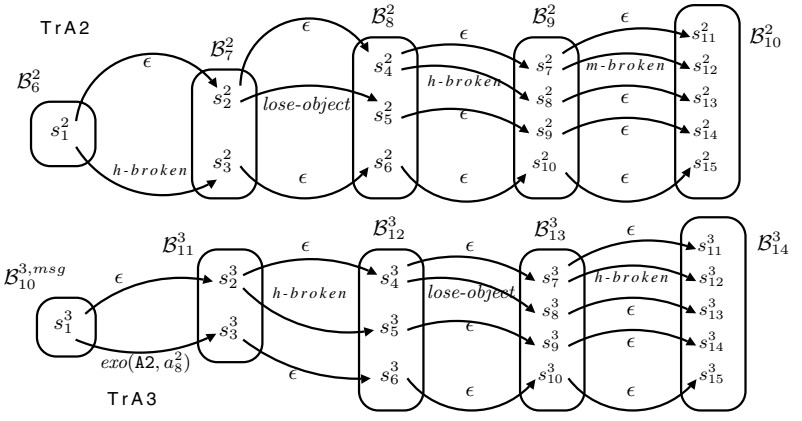


Fig. 2. The trajectory-sets computed by agents A2 and A3

Running example. Figure 2 shows two trajectory-sets, $TrA2$ and $TrA3$, built during the plan execution by agents A2 and A3 respectively; these trajectory-sets are consistent with the observations received by the agents so far. Note that each rectangle encloses all the agent states within a given belief state. For instance, the belief state $\mathcal{B}_9^2$ includes the states s_7^2, s_8^2, s_9^2, and s_{10}^2.

The trajectory-sets keep trace of the possible occurrence of exogenous events during the execution of the actions: for example, the exogenous event *lose* means that the agent can lose a block during a *carry* action; whereas *m-broken* and *h-broken* refer to a fault respectively in the mobility and handling functionality of the agent, the first fault affects the *move* action, the second affects the *load*/*unload* actions.

More important, since there exists a causal link between actions a_8^2 and $a_1 1^3$, the agent A2 sent a message to agent A3 about the service AT(Om,L3) (see Figure 1). However, action a_8^2 has a pending outcome, thereby the message received by A3 is ambiguous and maintains a reference to an exogenous event, which possibly occurred during the execution of a_8^2; this ambiguous message has been consumed by agent A3 and included in the trajectory-set $TrA3$ through the two transitions between the belief states $\mathcal{B}_{10}^{3,msg}$ and $\mathcal{B}_{11}^3$: the transition labeled with ϵ models the accomplishment of service AT(Om,L3), the transition labeled with $exo(\texttt{A2}, a_8^2)$ models the occurrence of an exogenous event.

5 Cooperative Plan Monitoring and Diagnosis

In this section we discuss how the trajectory-set inferred by agent i can be refined by exploiting pieces of information provided by other team members. The refinement of the trajectory-set is an important step, which allows agent i to determine the outcome of some pending actions in pO^i. To reach this objective, one has to single out which pieces of information should be exchanged among the agents and when.

agent i
procedure PropagateSuccess(a_l^i) {
  for each link $cl \in T_{in}^i | cl : a_m^j \xrightarrow{q} a_l^i$
    notify agent j:
      service q in $cl : a_m^j \xrightarrow{q} a_l^i$ accomplished
}

agent j
procedure ExploitSuccess(q, cl) {
  assert q in $\mathcal{B}_{m+1}^j$ and prune Tr^j
  assert outcome(a_m^j) = *succeeded*
  revaluate outcome for each action in pO^j
}

Fig. 3. The procedures for success propagation

In principle, the agents could exchange one another the observations they receive from the environment. However, this approach may suffer from two drawbacks: first, the amount of data the agents need to exchange could be too large; second, the data concerning the observations received by an agent could not be necessarily useful for another agent.

As discussed in the previous section, in our approach the agents exchange the results of a local interpretation process aimed at inferring the action outcome. In particular, the messages are sent not only when an agent detects the success or failure of an action, but also in presence of ambiguity (pending outcome). In the following of the section, we will show how the ambiguous results provided by agent j to agent i can be refined on the basis of the feedback provided by agent i, so that agent j can determine the outcome of some previously pending action.

First of all, let us consider the case when agent i detects the nominal outcome for action a_l^i on the basis of the observations $obs(a_l^i)$. In this case, the trajectories in $Tr^i[0..l]$ resulting from the monitoring step involve only ϵ transitions, and the agent i can conclude that the execution of all previous actions is nominal and all the services needed for action a_l^i have been provided; in this way it is possible to determine the outcome of some pending action in pO^i. Moreover, the nominal outcome of action a_l^i provides a positive feedback to agent j, in fact the following property assures that at least the nominal outcome of action a_m^j is detected.

Property 1. Let a_l^i be an action in A^i, and let a_m^j an action in A^j such that there exists a causal link $cl \in T_{in}^i$, $cl : a_m^j \xrightarrow{q} a_l^i$, and let us assume the outcome of action a_m^j is pending; if a_l^i has outcome *succeeded* then q has been certainly accomplished and a_m^j has outcome *succeeded*.

For this reason agent i invokes procedure **PropagateSuccess** (see Figure 3) to notify other agents about this outcome. Observe that the propagation is performed by considering only the subset of agents which provide action a_l^i with some service q (see the incoming causal links in T_{in}^i).

The message about the accomplished service q is exploited by agent j, by invoking procedure **ExploitSuccess**, whenever the agent j is not sure to have provided q. To consume this message, the agent j asserts the atom q within its trajectory-set Tr^j; more precisely, since the atom q refers to the effects of action a_m^j, q must be asserted in the agent belief state inferred after the execution of a_m^j, namely $\mathcal{B}_{m+1}^j$. It is worth noting that as a side effect, the trajectory-set Tr^j can be refined by pruning off all those agent trajectories which are not consistent with q. Therefore, after this first step, the agent j reconsiders each pending action

agent i
procedure BackPropagateFailure (a_l^i){
infer local diagnosis D^i
let exist $cl \in T_{in}^i | \; cl : a_m^j \xrightarrow{q} a_l^i$
if the only explanation in D^i refers to $exo(j, a_m^j)$
notify agent j service q has not been provided
}

agent j
procedure ExploitBackPropFailure(q, cl){
assert $\neg q$ in $\mathcal{B}_{m+1}^j$ (pruning of Tr^j)
assert outcome(a_m^j) = *failed*
revaluate outcome for each action in pO^j
infer local diagnosis D^j
}

Fig. 4. The procedures for back propagation of an action failure

in pO^j, as it is possible that the trajectory-set Tr^j has been sufficiently refined to determine the outcome of some of them.

In case agent i detects the failure of action a_l^i, a local diagnosis process is immediately activated to provide some possible explanations for that failure. A local diagnosis can be inferred directly from the trajectory-set as follows.

Definition 6. *Given the failure of action a_l^i, and the trajectory-set $Tr^i[0..l]$, the* local diagnosis *for that failure is* $D^i =$ PROJECTION $_{e_0,\ldots,e_l} Tr^i[0..l]$.

In other words, the local diagnosis for the failure of a_l^i is a set of sequences of events, where each sequence *seq* has the form $\langle e_0, e_1, \ldots, e_l \rangle$ and represents a possible explanation. Each event e_k $(k : 0..l)$ is in $\mathcal{E} \cup \{\epsilon\}$, however, since a not nominal outcome has been detected, in each sequence *seq* at least one anomalous event must be occurred.

It is important to note that some of the explanations included in the local diagnosis could refer to anomalous events concerning services provided by other agents; that is, it is possible to explain the failure of action a_l^i as an indirect consequence of the failure of some actions performed by other agents.

In particular, when agent i is able to explain its local failure just as a consequence of a failure in the local plan by agent j, it invokes the procedure **BackPropagateFailure** (see Figure 4), to notify agent j that service q has not been provided. Whenever agent j receives such a message activate procedure **ExploitBackPropFailure**: the local trajectory-set of agent j is refined by asserting $\neg q$ in $\mathcal{B}_{m+1}^j$; after this step it is therefore possible for agent j to determine whether other actions, besides a_m^j, are failed, and hence the outcome for each pending action in pO^i is evaluated again. Finally, agent j infers the local diagnosis D^j according to definition 6.

Running example. Let us consider again the trajectory-sets in figure 2, and assume that after the execution of action a_9, agent `A2` receives the message *"Position equals L2"*. This piece of information is used to prune the trajectory-set of `A2`, which is able to conclude that action a_9 has outcome *succeeded* (the agent knows that it has reached position *L2* as exepected). However, `A2` does not know whether the object `Om` has been delivered to agent `A3` or not, thus its set of pending outcomes is $pO^{\texttt{A2}}$=$\{a_6, a_7, a_8\}$.

Now, let us assume that the set of pending outcomes for agent `A3` is $pO^{\texttt{A3}}$= $\{a_{11}, a_{12}, a_{13}\}$, but after the execution of action a_{13} the agent `A3` receives the observation *"Object Om on top of object Oh"*, also in this case the observation

is used to prune the trajectory-set of A3, in particular the result of the pruning consists in removing all the anomalous trajectories, and as a consequence A3 determines the nominal outcome for each action in $pO^{\mathtt{A3}}$. Observe that, as soon as agent A3 determines the nominal completion of action $a_1 1$, depending on services provided by A2, it notifies A2 that those services have been provided. In fact, the nominal outcome of action a_{11} implies also a nominal outcome for action a_8, which in turn implies the successful completion also for action a_6 and a_7.

6 Discussion and Conclusion

In recent years increasing attention has been devoted to plan execution and in particular to plan diagnosis ([1,6]). In fact, the early detection of an action failure, and of a possible explanation of its causes, are essential to start a recovery step (see e.g. [4]). The framework by Roos et al. [2] has many similarities with the framework we propose as it considers a precise notion of multi-agent plan, where actions are atomic and concurrently performed by a team of agents. However, Roos et al. discuss a centralized approach: the diagnostic problem takes into account all the agents in the team and all the available observations at a given time. Moreover, their action models consider just the nominal action behavior, while any abnormal behavior is *unknown*. On the contrary, the framework we have discussed is distributed: each agent is responsible both for executing actions and for diagnosing them. In addition to that, we can model the nominal as well as the anomalous behavior of an action; in particular, the anomalous behavior may be non deterministic or even abstracted by *unknown*.

It is important to note that the complexity of the plan diagnosis task strongly depends on the amount of observations available to the agents. In previous works ([3]) we have described methods able to detect and diagnose action failures when each agent has sufficient information for certainly detecting the outcome at least of each action providing other agents with services.

In the present paper we have considered a very challenging scenario where the degree of observability is so low, that an agent may not determine the outcome of an action even when that action provides services to other agents. A demanding consequence of such a scenario is that the agents must be able to deal with ambiguous action outcomes and need to communicate one another to refine as far as possible their beliefs. To face this problem we presented the weak-commitment policy, which allows *pending* outcomes to be turned into success or failure through the exchange of messages among agents. Notice that messages are not raw data (such as low-level observations), but are action outcomes. The advantage of this solution is twofold: first, agents reduce the amount of data to be communicated; second, agents exchange much more informative data since action outcomes are the results of (local) interpretation processes.

A preliminary set of experiments is currently carried on for testing the approach, which has been implemented by extending the software prototypes used in [3], and by exploiting the symbolic formalism of the Ordered Binary Decision Diagrams for compactly encoding both action models and trajectories.

References

1. Kalech, M., Kaminka, G.A.: On the design of coordination diagnosis algorithms for teams of situated agents. Artificial Intelligence 171(8-9), 491–513 (2007)
2. Roos, N., Witteveen, C.: Models and methods for plan diagnosis. Journal of Autonomous Agent and MAS 19(1), 30–52 (2009)
3. Micalizio, R., Torasso, P.: Monitoring the execution of a multi-agent plan:dealing with partial observability. In: Proc. of ECAI 2008, pp. 408–412 (2008)
4. Micalizio, R.: A distributed control loop for autonomous recovery in a multi-agent plan. In: Proc. of the 21st IJCAI 2009 (to appear, 2009)
5. Cox, J.S., Durfee, E.H., Bartold, T.: A distributed framework for solving the multiagent plan coordination problem. In: Proc. AAMAS 2005, pp. 821–827 (2005)
6. Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. In: Proc. Int. Conf. on Autonomous Agents (ICAA 2001), pp. 529–536 (2001)

Strategies for Exploiting Trust Models in Competitive Multi-Agent Systems

Víctor Muñoz, Javier Murillo, Beatriz López, and Dídac Busquets

Institut d'Informàtica i Aplicacions
Campus Montilivi, edifice P4, 17071 Girona
{vmunozs,jmurillo,blopez,busquets}@eia.udg.edu

Abstract. Multi-agent systems where agents compete against one another in a specific environment pose challenges in relation to the trust modeling of an agent aimed at ensuring the right decisions are taken. A lot of literature has focused on describing trust models, but less in developing strategies to use them optimally. In this paper we propose a decision-making strategy that uses the information provided by the trust model to take the best decisions to achieve the most benefits for the agent. This decision making tackles the exploration versus exploitation problem since the agent has to decide when to interact with the known agents and when to look for new ones. The experiments were performed using the ART Testbed, a simulator created with the goal of objectively evaluate different trust strategies. The agent competed in and won the Third International ART Testbed Competition held in Estoril (Portugal) in March 2008.

Keywords: Competitive multi-agent systems, Trust, Reputation, ART Testbed.

1 Introduction

There are, nowadays, various multi-agent environments in which agents act independently and compete to obtain the maximum benefits for themselves. These environments are usually composed of a large number of self-interested agents which interact with each other, offering and requesting services, in order to improve their performance. In order to decide which agents to interact with, agents generally use a built-in model of the other agents. This model gives the agent the information it needs to know which agents to *trust* in order to accomplish its objectives, as well as which agents to avoid. This is called the trust model of the agent [12].

The way an agent uses the trust model is similar to the way a human does. For instance, let us imagine a person, called *John*, who wants to buy a television. Last year John bought a television at the *Cheap & Good Store*, but shortly afterwards the television broke down. Consequently, John now has little trust in the *Cheap & Good Store*, and when it comes to buying another television he will prefer to buy it at another store. On the other hand, a relative of *John*, *Joe*,

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 79–90, 2009.

bought a television that worked well at the *Best TV Store* and he is therefore very pleased with the purchase and has a lot of trust in the store he bought it from. *Joe* mentions this to *John* who, as he trusts his relative, also has more trust in the *Best TV Store* (although *John* himself has not had any interaction with it). Finally, the *Best TV Store* can make publicity of itself, for example through advertisements.

As seen in the example, three different kinds of interactions can change the trust of an agent: (i) *direct trust* is based on personal experience, (ii) *indirect trust*, also called reputation, is based on another's experience of a third party, and (iii) *self trust* is based on an agent's advertising of itself. Although direct trust is in principle the most accurate, it is generally achieved at a higher cost (in terms of money, time, etc.) compared with the other kinds of trust [5]. Therefore, the use of indirect and self trust is indispensable in large environments in which it is not always possible (or it is too costly) to obtain direct trust. In such competitive environments, however, the use of indirect and self trust may be harmful, since some agents can act malevolently, given that a loss for a competitor may imply a benefit for itself. In fact, as a result of competitiveness it is quite likely that a considerable number of agents will not be honest, and try to deliberately produce losses in other agents in order to benefit from that behavior [16]. There is a distinction, though, between acting malevolently and giving bad quality service. To follow on from the previous example, a particular store may sell products of a higher quality than another, but this does not necessarily mean that the other store is acting malevolently. Conversely, a person may have a bad opinion of a particular store based on a bad past experience, and another person may have a good opinion of the same store based on a good past experience. If a third person asks them their opinions of the store, they will give opposing answers, but neither of them will be lying.

Therefore, the key requirement if an agent is to perform well is for it to have a trust model that takes into account all the above factors. However, a perfect trust model is difficult to obtain, and therefore the design of a strategy to take correct decisions based on an incomplete trust model (which may contain mistakes) is also an important factor if the agent is to obtain maximum benefits. In the decision-making process, the agent has to deal with the exploration versus exploitation problem [15], because when it wants to request a service it must decide whether to request it from one of the known agent providers (of services), or to explore some unknown ones and possibly discover better provider agents. If the agent is only focused on exploitation it might obtain good short-term results but bad ones in the long term, since exploration allows the agent to discover better provider agents than its normal providers, or to have substitute providers in case one of the usual providers fails (i.e. leaves the market or decreases in quality), a circumstance that is quite likely to happen when the agent is working in dynamic environments. Furthermore, exploration allows the agent to adapt to changes faster, given that if the quality of a provider diminishes for some reason, the agent will be able to change to another. Therefore, as an agent is usually limited in the number of interactions it can perform (due to time limits

and interaction costs), it is necessary to decide which interactions to dedicate to exploration, and which to exploitation.

In this paper we assume that a trust model has been previously selected and we present a strategy to use it, dealing with the exploration versus exploitation problem. To test the system, the ART (Agent Reputation and Trust) Testbed simulator is used [4]. The aim of this testbed is to provide a standard problem scenario which can be used to compare different approaches to modeling and applying trust and strategies in multi-agent systems.

This paper is structured as follows. In the following section we survey existing trust models available in the literature. Next, the strategy for using it is designed. Later, we use the ART Testbed simulator to test the strategy. Finally, some conclusions are given.

2 Related Work

Trust has been defined in different ways for different domains. For example, one of the most frequently used is the definition of trust as "confidence in the honesty or goodness of an agent". In this case trust is measured by the behavior of an agent, based on a decision about whether it is acting honestly or maliciously. However, the definition in [5] is the most useful for our purposes: "Trust is the confidence or belief in the competence, utility or satisfaction expected from other agents concerning a particular context". Under this definition, the trust in an agent to provide a specific service can be understood as an estimator of the expected quality that will be obtained from that service. For example, an agent having a trust level of 1 for one provider and a trust level of 0.5 for another provider (on a scale from 0 to 1, with 0 being the lowest trust level), the agent would prefer to interact with the first one since the expected quality would be higher.

We can find a good number of trust models in the literature, reviews of which can be found in [11]. Trust models usually incorporate only direct and indirect trust interactions, as for example REGRET [13], which calls *individual dimension* the direct experiences and *social dimension* the indirect ones. This work introduces the ontological structure concept, that is, the reputation is considered as a combination of different features (for example, the reputation of an airline company can be composed of a set of features like delays, food quality, etc.) from where the overall reputation value is computed after assigning weights to the individual components. However, there are some approaches as in [1], where a trust model based just on a reputation mechanism is proposed. Other works, like [5], use three components of trust: direct, indirect and self. Conversely, FIRE [7] presents four different kinds of trust: "interaction trust" which is equivalent to direct trust, "witness reputation" which matches indirect trust, "certified reputation" which is similar to self trust besides the fact that it is not the agent itself who provides this trust but its partners (the agents more related to it), and finally "role-based trust" for the trust associated to the set of agents belonging to the same organization, and applied to all of its members.

Other differences between the models concern the method used to manage the information extracted from interaction, i.e. how the incoming information is added to compute the trust model or update it. Most of the methods use aggregation functions, as in [5,19]. Others are based on probability theory [8,17], information theory [14], or the Dempster-Shafer theory [18], among others. In order to work with dynamics (agents that change their services quality through time) and incomplete information some works use a *forgetting function* that enables the selection of the most relevant information to build the trust model, as in [5] or [7].

More focused on the ART domain (explained in the results section), we also find a large number of papers in which domain-dependent trust models and strategies have been designed with the aim of participating in different international competitions. In [9], as in many other ART agents, the agent design is divided in three parts: the strategy for modeling other agents (trust model), the requesting strategy and the response strategy. The winner of the 2006 and 2007 ART international competitions, known as IAM [16], consists of three main parts: a lie detector to detect malicious agents, a variance estimator to estimate the quality of the other agents, and an optimal weight calculator to measure its own behavior against others. A later improvement of the IAM agent presented a trust model based on Bayesian reinforcement learning [15]. Another technique that has been used to design ART agents in order to deal with untrustworthiness is fuzzy logic. In [2], fuzzy logic is used to normalize the information received from each agent.

3 Using the Trust Model

Although there is a lot of work about trust models, there is less about how to use them. So, here we present a strategy for using a trust model. The trust model can be any that provides the following features:

- The direct, indirect and self trust of the agents, for each of the services that they offer, with a normalizable value between 0 and 1.
- The knowledge degree of a provider agent about a service (based only on direct trust). This is a value (normalizable between 0 and 1) that represents how much the agent has directly interacted with a provider, with 0 meaning that the provider is totally unknown (the agent has never directly interacted with that provider), and 1 totally known.

The information that the trust model provides is used by the agent to take decisions regarding service requesting. It is important to note that even with a very complex model of trust and a good strategy, if the information contained in the model does not correspond to reality or is incomplete, the agent will not be able to take advantage of it. For example, an agent that does not know all of the agents will not be competitive against another agent that does know all of them, even if the latter has a worse trust model than the former.

The decision-making process involves the resolution of the exploration versus exploitation problem, given that the number of interactions to perform is usually

limited. The key point here is the need to decide at every moment how much effort to spend on exploring for new unknown providers, and how much on exploiting the known ones.

In order to make this possible, our strategy is based on grouping the agents in four categories, according to the information stored about them in the trust model regarding a given service. Thus we consider the following categories:

- **Group TK (Totally Known agents):** The agents with a knowledge degree equal to 1. The trust model says that they are well-known providers since we have interacted with them many times.
- **Group PK (Partially Known agents):** Agents with a knowledge degree lower than 1 and greater than 0. These providers would probably become future providers in the event any of the existing ones failed.
- **Group AU (Almost Unknown agents):** Agents with a knowledge degree equal to 0. These are the agents without any direct interaction, but for which there is indirect or self information.
- **Group TU (Totally Unknown agents):** These are the agents without any information about, either direct, indirect or self.

Note that the membership of agents in the different groups changes through time, with all the agents belonging to the last group (TU) at the beginning. Moreover, agents that belong to the group TK may switch to lower groups if the trust model forgets old interactions (for dynamic environments).

These categories are used to define our exploration versus exploitation strategy. The total number of interactions T is limited due to time and cost restrictions, depending on the domain. Given a new set of services to be fulfilled, the agent exploits a set of M providers and explores a set of N agents, so that at the end the number of interacted agents is $T = M + N$.

3.1 Exploitation

In order to select the agents for exploitation, it is preferable to select from the providers belonging to the first group (TK), the ones that the agent thinks will provide good quality (direct trust higher than a given threshold QT, parameter that has to be set depending on the domain and the trust model) in supplying the requested service, as they are the most reliable agents. The services provided by these agents can be trusted since the agent has satisfactorily interacted with them many times in the past, and consequently it is quite likely that future interactions will be successful as well. In the case that there are not sufficient agents conforming to the above restriction, the agent can use as providers the rest of the agents from the first group, or the best agents from the second group (at a higher level of risk). As a result of the exploitation, the parameter M will be fixed (the number of agents with a direct trust higher than QT, with an upper limit of T).

3.2 Exploration

The main idea of the exploration process is to gradually raise agents from lower to higher groups until they conform to the first group's constraints. It is not

mandatory to use exploration, but its benefits are clear, since without it, if the usual providers fail (the providers become unavailable or their quality decreases too much), the agent would have to start searching, perhaps interacting with totally unknown (potentially risky) agents in order to find new providers. Instead, choosing correctly the agents for exploration will move more agents to the known category, thus allowing the agent to have candidates for future exploitation, in the event that any regular providers are lost.

The exploration process could be performed by choosing the unknown agents randomly. However, we have designed a strategy that achieves a better outcome (as demonstrated in the results section 4.4). This mechanism consists of three phases and its objective is to optimally spread the exploration interactions of the agent. The agents to explore are taken from the groups PK, AU and TU, with the available interactions ($N = T - M$) being distributed according to the three following phases:

1. Select agents from the PK group. Agents are sorted in descending order according to their direct trust regarding the service to be covered, and only the best are selected. The objective of this phase is to know completely these agents, in order to move them to the TK group.
2. If there are still interactions left, the agents are taken from the AU group. The agents in this set are arranged in descending order according to their indirect trust, and only the best are selected. A single interaction is assigned to each provider, until exhausted. These agents will belong to the PK group in the next time step.
3. Finally, the agents are selected from the TU group, and a single interaction is performed in order to move them to the AU group. Here, the providers are selected at random, as we do not know anything about them.

The different phases of the mechanism are subsequently executed until the available exploration interactions (N) are exhausted.

3.3 Initial Time Steps Procedure

In the initial time steps no trust model has yet been built, and indirect trust is not useful either, because initially nobody knows anybody, so any received reputations would not be right. Therefore, the strategy here is to use the self trust values obtained from the agents to decide which agents to rely on.

Another strategy for the initial time steps is to trust the agents that answer our requests, and the agents that make requests to us. This is because we expect that agents interested in interacting with us will be truthful agents, while untruthful agents are generally less interested in interacting with others.

3.4 Agent Behavior for Service Providing

Up to now we have discussed how our agent requests services to others (requesting services). Here we talk about strategies for the agent behavior with the other agents' requests (providing services).

We believe that giving bad services qualities deliberately is not beneficial since in the long term the other agents either stop trusting, stop requesting, or even start giving bad qualities to us. Thus, we lose possible providers. Moreover, in dynamic environments, an agent that at a given time is not a good provider can become one in the future. Therefore, acting malevolently would produce a reaction against us in the agents, so we decide to act always honestly.

Finally, with regard to reputation requests, we decided to answer them truthfully. In doing so, the agents with good trust models (models that make use of indirect trust) are favored over the rest, and we expect them to answer back in the same way, thereby achieving mutual collaboration.

4 Experimentation

In this section we explain the tool used to test our work: the ART Testbed, and the results obtained on the ART Testbed international competition.

4.1 ART Testbed

The Agent Reputation and Trust Testbed[1] is a simulator for the art appraisal domain, "a working framework created with the goal of serving as an experimentation tool of different trust models and strategies for them, as well as a forum in which researchers can compare their technologies against objective metrics" [4]. It simulates an environment in which the participant agents act as art *appraisers* that compete to obtain the most clients. Appraisers receive more clients, and thus more profit, for giving more accurate appraisals. The simulator generates a fixed total number of clients that request appraisals for paintings that belong to different artistic eras (e.g. realism, impressionism, expressionism, etc.). The appraisers have varying levels of *expertise* in the different eras, making them experts in some but ignorant about others. This expertise can vary through time. During the game, the simulator progressively modifies the appraisers' client share according to the quality of their appraisals.

For the agents to perform the appraisals, they can ask other agents for their opinions (especially for the eras in which they are not experts) and set a weight for each of them in the simulator (expected to correspond to the agents trust). The final appraisal is then computed by the simulator as the mean of the agents' weighted appraisals. Each kind of interaction implies a payment from the requesting agent to the requested agent of a given amount of money, independent of the quality obtained in the interaction. The payment costs are more expensive for direct interactions than for other kinds of interactions. When the agent receives opinion requests from other agents, it has to decide how much money to spend on the opinion (the quality of the opinion depends on the amount of money spent). This money represents the appraisal effort made by the agent.

In addition to conducting opinion transactions (direct trust), appraisers can exchange *reputations*, or information about the trustworthiness of other appraisers (indirect trust). The self trust in this domain is called *certainty*. An important

[1] http://www.art-testbed.net

feature of the ART Testbed is that it is not necessary for the agents to provide good information. For each agent's request they can decide whether to cooperate or to act maliciously.

4.2 ART Agent

Figure 1 shows a simplified diagram of the interactions of our agent within the ART Testbed simulator (only direct interactions are drawn). First, the simulator sends the paintings to the agents (appraisal requests). The agent has to decide which agents to request an opinion from about each painting. As a result, the agent returns back the simulator a weight for each agent and era. The weight for an agent x and era j represents the importance that the agent wants to give to agent x's opinion for the paintings appraised of era j. Finally, the simulator calculates the final appraisal of the paintings and performs the weighted sum (with the weight given by the agent) of the appraisals requested. At the following time step, the agent will know the result of the appraisals given by other agents since the simulator sends the true values of the paintings.

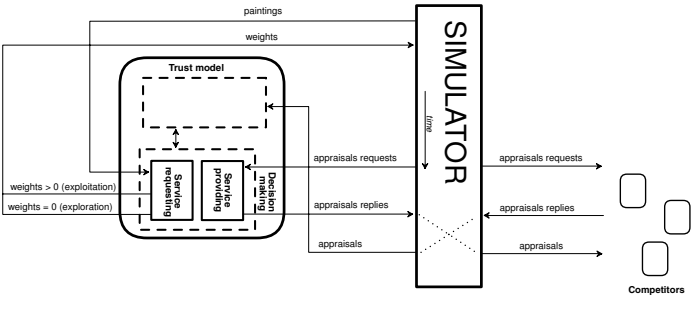


Fig. 1. Interactions with the ART Testbed simulator

In this domain the agent can simulate exploration and exploitation with the weights, by giving a weight greater than zero for exploitation, and equal to zero for exploration. When a weight equal to zero is given to the simulator the request is performed, so the agent will later know the appraisal given by agent but this will not affect the final appraisal.

The amount of effort dedicated to exploitation (M interactions) is given by the quality threshold QT, so that agents with a trust value higher than this threshold will be used as appraisers, with a limit of T. This defines the maximum number of questions that the agent wants to perform for each painting in order to spend a given percentage ($questionPercentage$) of the earnings of an opinion ($\frac{clientFee}{opinionCost}$). This limit is calculated according to Equation 1:

$$T = \min\left(\frac{clientFee \cdot questionPercentage}{opinionCost}, maxNbOpinionRequests\right) \quad (1)$$

where *clientFee*, *opinionCost* and *maxNbOpinionRequest* are parameters of the simulator representing, respectively, the price earned at each transaction, the cost of a requested opinion, and the maximum number of opinion requests that an agent can send for each painting at each time step. Alternatively, the parameter *questionPercentage* is agent's own and indicates the percentage of the earnings that it wants to spend on asking other agents.

During the exploitation selection M interactions have been performed, if there are still interactions left (if $M < T$), the rest ($N = T - M$) will be used in the exploration process, following the phases previously explained in Section 3.

4.3 Exploration Algorithm Evaluation

With the aim of evaluating our exploration procedure for the selection of agents to be discovered in the ART Testbed domain, the following experiment was designed. We created a game with 10 copies of our agent equipped with the trust-based exploration mechanism previously explained. We also added to the game ten copies of the agent with a random selection of candidates to explore. The trust model, the exploitation mechanism and the parameters were the same for the two groups of agents. We also added dummy agents to the game: five dishonest (act malevolently) and five honest. The parameters of the game were the following: 100 time steps, 20 eras, 4 $numberOfErasToChange$. The rest of the parameters were set to their default values.

In order to obtain a metric for the comparison we defined the useful information index (UII), which measures the useful information that the agent has found in the environment. This metric is defined as follows:

$$UII = \frac{\sum_{i=0}^{i<numEras} GA(i)}{numEras} \tag{2}$$

where the value of $GA(i)$ is 1 when the agent has two or more agents with a quality in the era i higher than the quality threshold (QT). The value is 0.5 if there is only one agent satisfying this condition and 0 if there are no agents.

Figure 2 shows the UII average of the ten agents with the trust-based exploration mechanism and the UII average of the ten agents with the random mechanism. We can see that a stabilization phase appears during the 20 first time steps. During this phase the agents get to know the environment until they arrive at a stabilization point. From that point, the UII value oscillates due to the expertise changes produced in the environment. During the stabilization phase, the UII grows faster and reaches a higher maximum value in the agents with the trust-based exploration strategy than with the random mechanism. Furthermore, the value reached can be maintained at this higher value. Therefore, we conclude that the designed strategy allows the agent to feed the trust model with more useful information than a random method.

4.4 Competition Results

We now analyze the performance of the agent compared with other real agents. We designed an agent named Uno2008 that uses a trust model similar to [5] with

Fig. 2. Comparison between the trust-based exploration mechanism and a random exploration mechanism

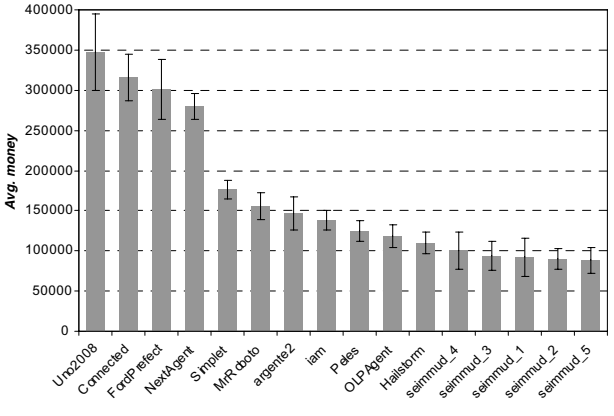


Fig. 3. Average and standard deviation of the earnings in the 2008 ART Competition

some adjustments. Due to space limitations we cannot explain the whole trust model (for a detailed explanation see [10]). The parameter QT has been set to 0.7, and $questionPercentage = 0.4$; these values have been found empirically, although the behavior of the agent does not change abruptly with similar values.

The results are taken from the 2008 International ART Testbed Competition held in Estoril, at AAMAS. In this competition 11 agents were registered from 7 different countries. Five dummy agents were also added. The competition consisted of three different kinds of games, the first with low dynamics (# eras to change (ETC) = 1, amount of expertise change (AEC) = 0.05), the second with medium (# ETC=3, AEC = 0.1) and the third with high dynamics (# ETC=3,

AEC = 0.3). Each was repeated three times, and the average of earnings of the agents in the three games was computed to determine the final scores.

The results are shown in Figure 3, where the y axis represents the average earnings obtained for each agent in all the games, with its standard deviation. Our agent, Uno2008, managed to win the competition by obtaining the highest score, with Connected and FordPrefect completing the podium. Agent Uno2008 won eight of the nine games played. We must also highlight the big difference in the first four players over the rest. The last five agents, called "seimmud", correspond to the dummy agents.

5 Conclusions

In environments with competitive agents, an agent can behave maliciously, trying to harm other agents, with the aim of obtaining better results. Alternatively, the agents can offer different service qualities. For these reasons, trust is a very important factor as it allows us to know about the behavior of agents and to predict the results of interactions with them, and consequently to make better decisions. However, a perfect trust model is difficult to obtain, and therefore the design of a strategy to take correct decisions based on an incomplete trust model (which may contain mistakes) is also an important factor if the agent is to obtain maximum benefits.

In this article, a strategy for using a trust model in a decision-making process has been presented. The required trust model must be based on three different trust components: direct, indirect and self. Direct trust is based on the agent's own experiences, indirect trust (reputation) is based on other agent's experiences, and self trust is the publicity that an agent transmits about itself. The data of an agent's trust model has to be processed in order for the best decisions to be taken and leading to its benefits being maximized or its objectives being obtained. The process of making the decisions involves the exploitation versus exploration problem. To solve this problem, we classify the agents in four categories (totally known, partially known, almost unknown and totally unknown). We use a method for exploration that combines the chance of finding good information in partially known agents with the random factor.

The design of the agent was tested in the ART Testbed domain. It participated in the 2008 international competition held in Estoril (in AAMAS), which it won. As future work, we are studying the possible application of these strategies in other real domains, such as Internet electronic service providers, to check whether they behave as well as they did in the ART competition.

Acknowledgments. This research project has been partially funded by the Spanish MICINN project SuRoS (TIN2008-04547) and with the support of the Commissioner for Universities and Research of the Department of Innovation, Universities and Company of Generalitat of Catalonia and of the European Social Background, and the University of Girona BR Grant program.

References

1. Abdul-Rahman, S.H.A.: Supporting trust in virtual communities. In: The 33rd IEEE International Conference on Systems Sciences, pp. 4–7 (2000)
2. Del Acebo, E., Hormazábal, N., de la Rosa, P.: Beyond Trust. Fuzzy Contextual Corrective Filters for Reliability Assessment in MAS. In: The Workshop on Trust in Agent Societies at AAMAS 2007, Honolulu, Hawaii, USA, May 15, pp. 44–48 (2007)
3. Falcone, R., Singh, M., Tan, Y.-H. (eds.): AA-WS 2000. LNCS (LNAI), vol. 2246, pp. 54–72. Springer, Heidelberg (2001)
4. Fullam, K., Sabater, J., Barber, S.: Toward a testbed for trust and reputation models. Trusting Agents for Trusting Electronic Societies, 95–109 (2005)
5. Gómez, M., Carbó, J., Benac, C.: Honesty and trust revisited: the advantages of being neutral about other's cognitive models. Autonomous Agent and Multi-Agent Systems 15, 313–335 (2007)
6. Gui, C., Wu, Q., Wang, H.: Towards Trustworthy Resource Selection: A Fuzzy Reputation Aggregation Approach. Autonomic and Trusted Computing 4610/2007, 239–248 (2007)
7. Huynh, T.D., Jennings, N., Shabdolt, N.R.: An integrated trust and reputation model for open multi-agent systems. In: AAMAS 2005, pp. 512–518 (2005)
8. Josang, A., Ismail, R.: The beta reputation system. In: Proceedings of the 15th Bled Conference on Electronic Commerce (2002)
9. Kafali, O., Yolum, P.: Trust Strategies for the ART Testbed. In: The Workshop on Trust in Agent Societies at AAMAS 2006, pp. 43–49 (2006)
10. Muñoz, V., Murillo, J.: Agent UNO: Winner in the 2nd Spanish ART competition. Revista Iberoamericana de Inteligencia Artificial 39, 19–27 (2008)
11. Ramchurn, S., Hyunh, D., Jennings, N.R.: Trust in multi-agent systems. The Knowledge Engineering Review 19, 1–25 (2004)
12. Sabater, J.: Trust and Reputation for Agent Societies, Monografies de l'institut d'investigació en intel.ligència artificial, 20, PhD Thesis (2003)
13. Sabater, J., Sierra, C.: Regret: A reputation model for gregarious societies. In: Fourth Workshop of Deception, Fraud and Trust in Agent Societies, pp. 61–69 (2001)
14. Sierra, C., Debenham, J.: An information-based model of trust. In: AAMAS 2005, pp. 497–504 (2005)
15. Teacy, W.T.L., Chalkiadakis, G., Rogers, A., Jennings, N.R.: Sequential Decision Making with Untrustworthy Service Providers. In: AAMAS 2008, pp. 755–762 (2008)
16. Teacy, L., Huynh, T.D., Dash, R.K., Jennings, N.R., Luck, M., Patel, J.: The ART of IAM: The Winning Strategy for the 2006 Competition. In: Proceedings of the 10th International Workshop on Trust in Agent Societies, pp. 102–111 (2007)
17. Teacy, W., Patel, J., Jennings, N., Luck, M.: TRAVOS: Trust and Reputation in the Context of Inaccurate Information Sources. In: AAMAS, vol. 12, pp. 183–198 (2006)
18. Yu, B., Singh, M.: A social mechanism for reputation management in electronic communities. In: Klusch, M., Kerschberg, L. (eds.) CIA 2000. LNCS (LNAI), vol. 1860, pp. 154–165. Springer, Heidelberg (2000)
19. Zacharia, G., Maes, P.: Trust management through reputation mechanisms. Applied Artificial Intelligence 14, 881–907 (2000)

A Distributed Detecting Method for SYN Flood Attacks and Its Implementation Using Mobile Agents*

Masaki Narita, Takashi Katoh, Bhed Bahadur Bista, and Toyoo Takata

Iwate Prefectural University, Iwate, Japan

Abstract. In recent years, damage caused by DoS attacks is real and causing substantive problems. Such threat is widespread from major commercial sites to individual users. Therefore, it is important for network administrators to develop means to comprehend the latest trend of DoS attacks. In this paper, we propose a distributed detecting method for SYN Flood attack which exploits a flow in TCP itself. Our proposed system employs mobile agents to detect SYN Flood attack. We also show the effectiveness of our proposal through experiment of detection of SYN Flood attack in virtual network of simulation environment.

1 Introduction

The Internet is essential not only for daily life but also for business these days. With such a situation, the risk for computers connected to the Internet being attacked is increasing.

One of the threats on the Internet is DoS (Denial-of-Service) attack. DoS attacks are malicious actions which place burden on network servers intentionally to bring down or hinder the services.

DoS attacks are recognized as serious social problems all over the world. The damage caused by DoS attacks has not decreased at all. On the contrary, DoS attacks have become large-scale and complicated threat. Furthermore, it is expected that this situation will continue in the future.

Financial damage of victim companies is also not negligible because DoS attack is used for the means of intimidation [1] as GovCert.nl (Dutch Computer Emergency Response Team) [2] suggests. Moore *et al.* show that such DoS threat is widespread from major commercial sites to individual users [3]. This situation means that anybody has possibility to be a victim at any time. Thus, it is important for network administrators to develop means to comprehend the latest trend of DoS attacks.

Moreover, nowadays users whose computers have been used for attacks without their knowledge come under scrutiny even though they are not attackers themselves. Because of this, to detect the latest trend of attacks promptly is meaningful for not becoming attackers unknowingly, too.

* This work was supported by Grant-in-Aid for Young Scientists (B) (21700084).

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 91–102, 2009.

In this paper, we propose a distributed detecting method for SYN Flood attacks. Our method is useful for network administrators because they can develop strategies for attacks and get chances to protect their network itself. For example, they can prevent generating DoS attackers unknowingly within their network.

The rest of this paper is organized as follows. Sect. 2 describes SYN Flood attacks which we aim to detect. Sect. 3 describes existing works regarding SYN Flood detection and their problems. In Sect. 4, we propose a distributed detecting method for SYN Flood attacks. In Sect. 5, we show the effectiveness of our proposal using simulation results. Finally, we conclude our work in Sect. 6.

2 SYN Flood Attacks

SYN Flood attack which we aim to detect is a kind of DoS (Denial-of-Service) attack. DoS attack is a malicious action which places burden on the network server intentionally to bring down or hinder the services.

DoS attacks can be classified by protocol into two types: TCP and UDP based attacks. Moore *et al.* studied that 90% of all DoS attacks are TCP based [3]. The type of SYN Flood attack which we aim to detect is TCP based attack.

In this section, we explain the procedure of TCP connection establishment before we refer to SYN Flood attacks. Then, we describe the mechanism of SYN Flood attacks and their features.

2.1 TCP Connection and Connection Establishment

TCP is an Internet standard protocol based on reliability. The procedure of TCP connection establishment is called 3-way handshake.

To explain the correct TCP establishment, we assume two hosts, host A and host B (Fig. 1). First, host A sends a SYN packet to host B to request establishing a connection. Host B replies with a SYN/ACK packet to host A to acknowledge connection request and to request establishing a connection in reverse. Finally, host A sends an ACK packet to host B to acknowledge connection request.

2.2 Mechanism of SYN Flood Attacks

SYN Flood attacks abuse TCP establishment procedure mentioned in the previous subsection. The overview of a SYN Flood attack is shown in Fig. 2.

An attacker sends a large amount of SYN packets whose source addresses are spoofed, to a victim host. The victim host does not have means to identify whether the source addresses of received packets are spoofed or not. Thus, the victim host responds to those spoofed addresses.

Each connection information is managed respectively in the TCP protocol. The victim host could expend all its listening queues just waiting for ACK from *source* hosts. In other words, the victim host has to maintain half-open connection to many irrelevant hosts on the Internet. The victim host is in danger of slowing down or crashing in the worst scenario. The slowdown of the host leads to degradation of service quality provided by the host and if it is crashed, it cannot keep providing any services anymore.

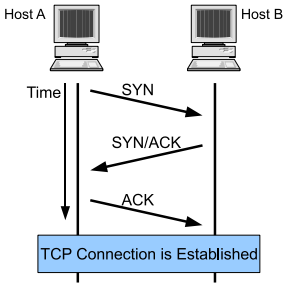


Fig. 1. 3-way handshake of TCP connection establishment

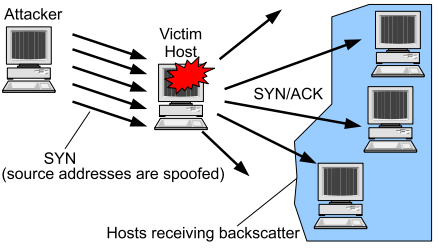


Fig. 2. Overview of SYN Flood attacks

2.3 Features of SYN Flood Attacks

Source addresses are spoofed in SYN Flood attacks as we mentioned above. Thus, it is possible that SYN/ACK packets arrive at irrelevant hosts abruptly. These packets are called backscatter. Catching these packets enables us to detect SYN Flood attacks.

Most attacking tools (e.g. [4,5]) spoof their source addresses randomly in a default setting. In that case, the backscatter will sparsely spread on the network. In this paper, we mainly detect SYN Flood attacks spoofing source addresses uniformly.

3 Related Works

3.1 Router Base SYN Flood Detection

There are some works regarding SYN Flood attack detection, e.g., [6,7]. Especially, Moore *et al.* monitored the class A network addresses for their research [3]. They defined the backscatter analysis and to quantify the DoS attacks for the first time in the world.

The router base SYN Flood detection methods monitor backscatter at a router. These methods, however, have following drawbacks:

- It is impossible to detect the attacks whose backscatter does not pass the router.
- Administrative access to the router is required. This means general users will have difficulties to acquire the information of SYN Flood attacks though it is important even for general users not to become attackers unknowingly.
- It is difficult to detect attacks if the inside network of the router is small because the number of backscatter which passes the router is almost proportional to the size of its network if the sources addresses of SYN packets are randomly spoofed.

3.2 Distributed SYN Flood Detection

There are some existing distributed systems.

DShield [8] collects the firewall log from volunteers all over the world. Monitored results are opened to the public on the web. It shows increasing accessed port number, and interestingly attacker's source IP address is also revealed.

@Police (National Police Agency, Japan) [9] places 57 network sensors in Japanese police facilities. They collect intrusion detection system log and firewall log. The result of analysis is updated per certain time as a graph and opened to the public on the web. @Police has the system specialized for monitoring backscatter to detect SYN Flood attacks [10].

When network administrators or individual users want to comprehend the latest trend of attacks, it is natural that they access to the web server which network monitoring organization provides. The information they get in this way is, however, a summarized result by such organizations per some time interval. Such a result may be an overview or fragmented information and it is difficult to acquire detailed information. In addition, such a result may not be the latest information.

From monitoring organization's point of view, revealing network sensors' addresses has a risk of being attacked. This implies they cannot reveal detailed information without careful consideration.

In recent years, a method to detect static sensors is devised [11]. Attackers can attack evading network sensors intentionally after detecting sensors using this method. Thus, there is possibility that the accuracy of results provided by monitoring organization decreases.

4 Proposal of a Distributed Detecting Method for SYN Flood Attacks

All the hosts connected to the Internet directly have the possibility of receiving backscatter. When some hosts receive backscatter, however, it is impossible to confirm whether similar backscatter is monitored in another network or not. Furthermore, one host cannot always receive sufficient amount of backscatter to detect trend of attacks because the backscatter spreads sparsely as mentioned above. Thus, it is difficult to detect the trend of attacks by only one host.

But if we can collect backscatter information among a number of distributed hosts, it will be possible to detect the trend of attacks. In this paper, we propose a distributed detecting method for SYN Flood attacks by collecting backscatter information among a number of distributed hosts.

4.1 Procedure of Our Method

The information which we can focus inside backscatter (a SYN/ACK packet to which ACK packets are not replied) is shown in Table 1. The source IP address and the source port number are identified from these information. We can comprehend the attacked host and attacked service by analyzing these two items.

Table 1. Information which we can focus inside backscatter

Source IP Address	Attacked Host
Source Port Number	Attacked Service
Destination IP Address	Spoofed IP Address
Destination Port Number	Port Number Used by Attacker
Acknowledge Number	Sequence Number Used by Attacker + 1

Our method consists of the following three steps:

1. Extract backscatter information (usually from log files created by network traffic monitoring software like `tcpdump`[1]) on each sensing point (a host or a router). The backscatter information consists of 1) the received time, 2) IP address and port number of source host (i.e., victim host), 3) IP address and port number of destination host, of the packet.
2. Collect these information from several sensing points. Each sensing point replies with the summarized information if it is requested for the information. The summarization is done by counting the number of backscatter packets for each source host which corresponds to the victim host of the SYN Flood attack at some time interval. Table 2 is an example of collected backscatter information using our proposal. In this example, time interval is 5 minutes. When counting backscatter, we also count the number of unique sensing points (destination hosts) which discover the backscatter generated by the identical attack to acknowledge how far the backscatter is spread on the Internet. We call this information *The Number of Unique Sensing Points*. If the total number of backscatter packets is near to the number of unique sensing points, we could assume that there was a SYN Flood attack whose source IP address is randomly spoofed.
3. Analyze the collected information.

The reason to collect the number of unique sensing points instead of the destination addresses themselves is that revealing network sensors' addresses (the destination addresses of backscatter) has a risk of being attacked.

4.2 Influence of Background Noise to Our Proposal

When SYN/ACK packets which are not backscatter arrive at a sensing point abruptly, they may become background noise which causes undesirable effect on the accuracy of detection result in our proposal.

Possible causes of such background noise are: (1) mis-configuration of a network device, (2) bugs in network software, (3) SYN/ACK scan, etc.

The cases (1) and (2) are not a big problem because we can easily get at the place of origin. Also, (3) SYN/ACK scan is very minor port scan method because attackers cannot acquire useful information by this scan. Thus, we expect that SYN/ACK scan is not used so frequently on the Internet.

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

Table 2. Example of collected backscatter information

Counting Time	Victim (Source) Host:Service	The Number of Packets
30 Dec. 2008 8:30	111.111.0.2:1025	5
	123.123.0.3:80	50
30 Dec. 2008 8:35	111.111.0.2:1025	10
30 Dec. 2008 8:40	111.111.0.2:1025	15
30 Dec. 2008 8:45	···	···
···	···	···

The Number of Unique Sensing Points	
111.111.0.2:1025	30
123.123.0.3:80	1

To confirm that such background noises are rarely generated, we investigated background noises arriving to one host having a global IP address. We collected traffic log on 2007/10/19–2007/10/22 and 2008/1/18–2008/1/21 (6 days) on the host.

From the result of our investigation, no background noise was found. Thus, we can conclude that we do not need to consider background noise too much in our proposal.

Even if background noise is generated in one sensing point, we could correctly judge those packets may be the result of the cases (1)–(3) using the number of unique sensing points. For example, we consider the case of visiting 50 hosts to collect backscatter information and 50 doubtful SYN/ACK packets were monitored as a result. If these 50 packets were monitored in about 50 unique sensing points, those packets information can be the correct result to detect the SYN Flood attack. On the contrary, if these 50 packets were monitored in only one sensing point, those packets information can be false positive generated by background noise.

4.3 Advantage of Our Proposal

As we mentioned in the previous section, router based detection has a difficulty to acquire the information of backscatter from wider range of network. It also has a difficulty for general users to acquire the information of backscatter.

Existing distributed systems for SYN Flood attack detection like @Police and CAIDA use static network sensors. Static network sensors monitor traffic at the same sensing points each time. Therefore, monitoring address range of network is restricted.

Our proposal enables us to monitor traffic anywhere on the Internet in principle because any host on the Internet can become a network sensor. Thus, even individual users can collect the latest trend of attacks in wider range.

5 Implementation and Evaluation of Our Proposal

We implemented our method and carried out an evaluation experiment to show the effectiveness of our distributed detecting method for SYN Flood attacks.

Because it is difficult to carry out SYN Flood attack on the real Internet for security reason, we carried out SYN Flood attacks inside the virtual network built on the network simulator. Then, traffic log obtained by the simulation is used for evaluation experiment. To collect backscatter information among a number of distributed hosts, we used ABLA (Agent Based Log Analyzing System) [12,13], which is a distributed Internet monitoring system.

In this section, we describe an overview of distributed Internet monitoring system ABLA. Then, we explain our implementation and evaluation experiment. Finally, we discuss our simulation result.

5.1 ABLA (Agent Based Log Analyzing System)

ABLA is a distributed Internet monitoring system developed under the GPL (GNU General Public License). The main reason we choose it as a base system is ABLA enables us to collect information from widely distributed hosts, that is difficult for router based systems.

In recent years, many organizations develop Internet monitoring systems. Most of those systems set up *static* sensors and *pile up* the data to one place periodically. After piling them up, analyzing such data is a usual procedure of most Internet monitoring systems.

On the contrary, ABLA is the Internet monitoring system which organizes a P2P network connected ABLA nodes with each other. The overview of ABLA is shown in Fig. 3. Each node in the ABLA network provides the network packets log for ABLA users. Each node becomes *dynamic* sensor. Mobile agents migrate among the ABLA nodes to *analyze* the packet log on each node.

This system is being developed for mainly individual users or small-scale network administrators. The ABLA users can detect the latest trend of attacks like new malware originating, and can make a countermeasure for the future attacks.

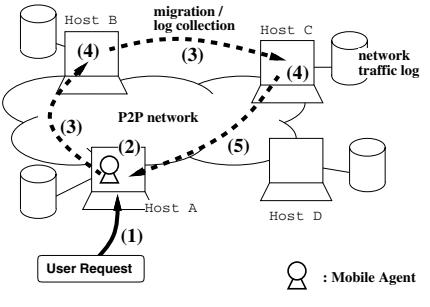


Fig. 3. ABLA overview

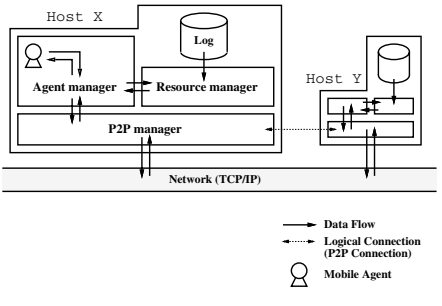


Fig. 4. ABLA architecture

Features of ABLA. ABLA has the following features.

Alleviation of Network Traffic: ABLA uses mobile agents to collect information. ABLA users collect information by giving a request to the agent. We do not need network resource to pile up data on one place, because analyzing task is done on each ABLA node. Thus, ABLA can alleviate the network traffic compared to other network monitoring systems.

Acquiring Dynamic Sensing Points: ABLA is an Internet monitoring system in which an individual user can participate for monitoring attacks. ABLA nodes can join or leave the ABLA network at any time. This enables us to acquire many dynamic sensing points. This method also solves the problem of evading static sensing points [11]. Additionally, ABLA can get more information than static sensors by increasing the number of ABLA nodes.

Establishing Anonymity: Anonymity of ABLA users is maintained all the time. We manage ABLA nodes using hash values generated by combination of IP address and using port number instead of IP address itself. There is no possibility of each ABLA node being mapped to its real IP addresses. Moreover, ABLA users can hide the information they do not want to reveal according to their security policy.

ABLA Architecture. The architecture of ABLA is shown in Fig. 4. We briefly review the role of each ABLA component and mobile agent as shown below.

Agent Manager: Agent manager generates mobile agents. When an agent is to be sent to another host, agent manager serializes the agent. When an agent manager receives an agent from another host, the agent manager deserializes and executes the agent.

Resource Manager: Resource manager manages the log information analyzed by each ABLA node. Resource manager analyzes the target log according to the request of a mobile agent and replies the result.

P2P Manager: P2P manager is responsible for establishing a P2P network. It is also responsible for making connection between two hosts via P2P network and sending / receiving serialized agents, and responsible for exchanging information of hosts that are directly connected by the P2P network (not physical connection).

Mobile Agent: Mobile agent is generated by an agent manager. Mobile agents migrate among the ABLA nodes according to the user request autonomously. Mobile agents issue a request to resource manager and collect an analyzed result of each ABLA node.

5.2 Implementation of Our Proposal

We implemented our method based on ABLA. We added a new module to the resource manager in the ABLA components, which manages the log information of each host, to implement a function for analyzing backscatter.

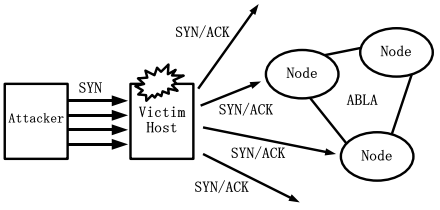


Fig. 5. Proposal overview

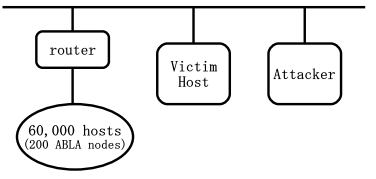


Fig. 6. Virtual network

This implementation allows ABLA users who want to detect SYN Flood attacks to request mobile agents to analyze backscatter information for detecting the trend of SYN Flood attacks. The overview of our proposal is shown in Fig. 5.

Generally speaking, it is difficult for general network administrators to get the latest trend of SYN Flood attacks enough because they cannot necessarily operate the large scale network monitoring system. Our proposal with ABLA enables general network administrators to collect the latest trend of SYN Flood attacks without such large scale network monitoring system.

5.3 Procedure of Evaluation Experiment

In the experiment, we generate specific patterns of SYN Flood attacks as shown in Figs. 7 and 9. We verify if we can grasp the time transition of attacks by changing the number of hosts mobile agents visit. To be more precise, we compare the analyzed result of an agent migrated and gathered in the monitoring network with the transition of all backscatter packets arrived at the monitoring network.

Evaluation experiment is composed of two steps.

Acquiring Attack Log: As the first step, we acquire attack traffic logs using network simulator. We use a customized version of 'Yet another network simulator' [14] Version 0.7.2 for our experiment. Virtual network is built and we generate SYN Flood attacks inside the network simulator. When backscatter arrives at the node on which ABLA is running, that backscatter is dumped to a file.

We built a virtual network like Fig. 6 to monitor backscatter when a SYN Flood attack whose source addresses are spoofed occurs. We made three subnets. The first one is monitoring network for backscatter. This network has 60,000 hosts including 200 ABLA nodes. The second and the third subnets have the victim host and the attacker host respectively.

Collecting SYN Flood Attack Information using ABLA: As the second step, we employ a simulation in which we issue an agent with a request according to our proposed scheme on the ABLA network. Each ABLA node has randomly assigned traffic log from the logs acquired by above-mentioned procedure.

5.4 Parameters of Evaluation Experiment

According to Moore *et al.* [3], 60% of all DoS attacks they monitored ended within 10 minutes and 85% of all DoS attacks ended within an hour. Therefore, we set the duration of SYN Flood attacks to 30 minutes in our experiment.

We assumed two attacking traffic patterns. One pattern is increasing attacking packets proceeded with time (Fig. 7). The other pattern is generating two rapid attacking within 30 minutes (Fig. 9).

60,000 hosts exist in the network simulator. 200 hosts are ABLA nodes in those hosts. We collected backscatter information from 25, 50, 75 and 100 ABLA nodes. We also changed an arriving rate of backscatter (a probability that a host receives backscatter packets). We used 0.50, 0.40 and 0.25 for arriving rates of backscatter. Thus $0.50n$, $0.40n$ and $0.25n$ (where $n = 60,000$) packets were sent from a victim host.

5.5 Results and Discussion

Our simulation was carried out 10 times changing collecting hosts of backscatter information randomly. The simulation result is an average value of every 5 minutes.

Result when Arriving Rate is 0.50 (Fig. 8 left, Fig. 10 left): When the backscatter information is collected from 100, 75 and 50 nodes, we obtained ideal detection results which are similar to the router traffic in both attacking traffic patterns. In the case of 25 nodes, we can also grasp the time transition of attack though attacking peak is a little vague in both attacking traffic patterns.

Result when Arriving Rate is 0.40 (Fig. 8 middle, Fig. 10 middle): When the backscatter information is collected from 100, 75 and 50 nodes, we obtained ideal detection results in both attacking traffic patterns. In the case of 25 nodes of attacking traffic pattern 1, it is also possible to grasp the time transition of attack. However, in the case of 25 nodes of attacking traffic pattern 2, we fail to detect the second attacking peak.

Result when Arriving Rate is 0.25 (Fig. 8 right, Fig. 10 right): When the backscatter information is collected from 100 and 75 nodes, we can grasp the time transition of attacks in both attacking traffic patterns though collecting packets are small. In the cases of 50 and 25 nodes of attacking traffic pattern 1, we fail to grasp the attacking peak. In the case of 50 nodes of attacking traffic pattern 2, first attacking peak is not detected. In the case of 25 nodes of attacking traffic pattern 2, because the line is flat, we must say we fail to detect the time transition of attack.

From these results, we can conclude that our method can properly detect SYN Flood attacks by gathering the backscatter information from wider range of network.

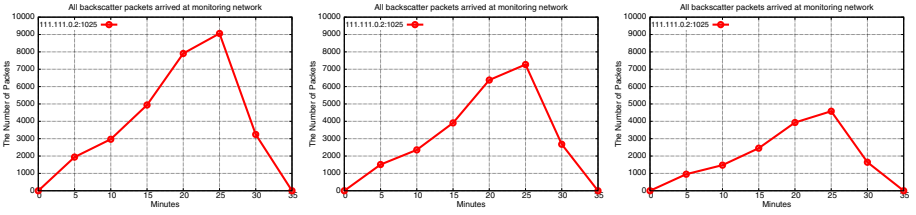


Fig. 7. Attacking traffic pattern 1 (Arriving rate: 0.50 (left), 0.40 (middle), 0.25 (right))

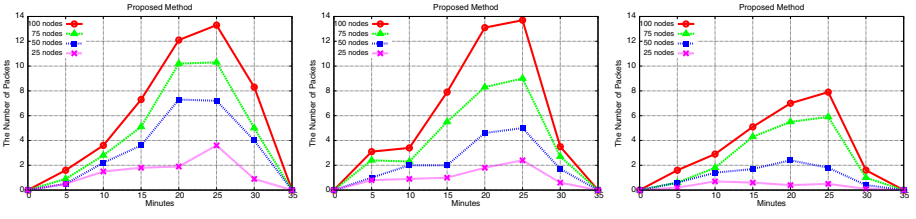


Fig. 8. Simulation result (Arriving rate: 0.50 (left), 0.40 (middle), 0.25 (right))

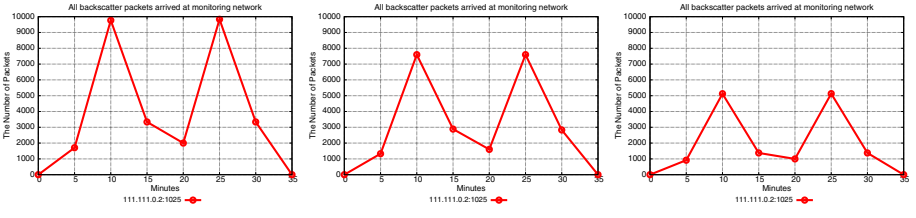


Fig. 9. Attacking traffic pattern 2 (Arriving rate: 0.50 (left), 0.40 (middle), 0.25 (right))

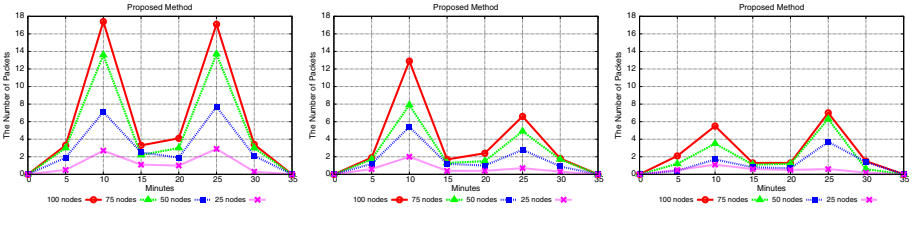


Fig. 10. Simulation result (Arriving rate: 0.50 (left), 0.40 (middle), 0.25 (right))

6 Conclusion

In this paper, we proposed a distributed detecting method for SYN Flood attacks by collecting backscatter information among a number of distributed hosts. We carried out simulation for detecting SYN Flood attacks using our method to prove the effectiveness of our proposal in the virtual network.

We obtained prospective results which is similar to the assumed traffic patterns as a consequence in the virtual network. Thus, we conclude our proposal is effective to collect the latest trend of attacks. This means our method enables small-scale network administrators to make a countermeasure according to the latest trend of attacks without depending on Internet monitoring organizations.

Our future work is evaluation of our proposal on the Internet. Developing new methodology for more detailed analysis is also our task, for example, automatic generation of an alert, guessing the attacking scale and so on.

References

1. SECURITY.nl: `http://www.security.nl/article/12088/1/Zombienetwerk_bestond_uit_1.5_milijoen_gehackte_computers.html`
2. GovCert.nl (Dutch Computer Emergency Response Team): `http://www.govcert.nl`.
3. Moore, D., Shannon, C., Brown, D.J., Voelker, G.M., Savage, S.: Inferring internet denial-of-service activity. ACM Transactions on Computer Systems (TOCS) 24(2), 115–139 (2006)
4. Stacheldraht: `http://staff.washington.edu/dittrich/misc/stacheldraht.analysis`
5. Synk4: `http://www.hoobie.net/security/exploits/hacking/synk4.c`
6. Kompella, R.R., Singh, S., Varghese, G.: On scalable attack detection in the network. IEEE/ACM Transactions on Networking 15(1), 14–25 (2007)
7. Wang, H., Zhang, D., Shin, K.G.: Change-point monitoring for the detection of dos attacks. IEEE Transactions on Dependable and Secure Computing 1(4), 193–208 (2004)
8. DShield: `http://www.dshield.org`
9. @Police: `http://www.cyberpolice.go.jp`
10. @Police: The system of monitoring syn flood attacks, `http://www.cyberpolice.go.jp/server/rd_env/pdf/synflood_detect.pdf`
11. Shinoda, Y., Ikai, K., Itoh, M.: Vulnerabilities of passive internet threat monitors. In: 14th USENIX Security Symposium (SEC 2005), pp. 209–224 (2005)
12. Katoh, T., Kuzuno, H., Kawahara, T., Watanabe, A., Nakai, Y., Bista, B.B., Takata, T.: A wide area log analyzing system based on mobile agents. In: Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce., November 2006, 7 pages (2006)
13. ABLA Project, `http://sourceforge.jp/projects/abla`
14. Yet Another Network Simulator, `http://yans.inria.fr`

Agent-Based Model for Decision Support in Multi-Site Manufacturing Enterprises

Zhan Sheng Ng[1], Aaron Yu Siang Tan[1], Arief Adhitya[2], and Rajagopalan Srinivasan[1,2]

[1] Department of Chemical and Biomolecular Engineering, National University of Singapore, 4 Engineering Drive 4, Singapore 117576, Singapore
[2] Institute of Chemical and Engineering Sciences, A*STAR (Agency for Science, Technology and Research), 1 Pesek Road, Jurong Island, Singapore 627833, Singapore

Abstract. Today's supply chains span across continents, involve numerous entities with different dynamics, and contend with various uncertainties. This paper presents an agent-based model for decision support in a multi-site lube additive manufacturing enterprise. The supply chain comprises raw material suppliers, the lube additive enterprise, and customers. The enterprise consists of a central sales department and a number of production sites at different locations. Each production site has its own commercial, scheduling, procurement, storage, operations, and packaging departments. Supply chain operation involves all these entities in three cycles of activities: enterprise-level coordination, plant operation, and inventory management. Each entity is modeled as an agent, with its own goals and tasks, implemented in Jadex following the belief-desire-intention (BDI) formalism. The model allows the user to simulate and analyze different supply chain policies, configurations, parameters, and scenarios. Its capability for decision support is illustrated through two case studies.

Keywords: Lube additive, agent-based modeling, simulation, supply chain, decision support, BDI.

1 Introduction

Lubricant (lube) additives are chemical products which enhance the performance characteristics of finished lubricating oils and greases. Additives are combined with base oil to produce formulated lubricant. Different types of additive perform different functions, for example, corrosion and rust inhibitors, anti-wear, anti-oxidants, anti-foams, friction modifiers, detergents to reduce buildup of deposit, etc. A lube additive package typically results from a complex formulation involving 10-15 ingredients. Altogether there could be over 4000 formulations from 1500 substances. Specific tailoring of lubricant composition is possible by using a different variety of base oils and additives. Hence, the key competence of additive suppliers is their ability to formulate unique additive packages that deliver required performance at competitive prices. Details of the intrinsic chemical identities and their proportions in the formulations are proprietary. Information about activities and relationships between component suppliers, formulators, customers is also confidential business information.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 103–114, 2009.

Interesting dynamics abound in the global lube additive supply chains. Chevron Oronite is one of the world's four largest suppliers of lube additives. On 11 August 2005, Oronite issued a global force majeure notice to seek relief from contractual requirements when performance is impossible due to events beyond their control. This tipped the marine lubricants industry into crisis, creating a serious situation which would affect deliveries of marine lubricants in ports in the Asia Pacific region. Oronite blamed rising global demand and tightened supply of raw materials. Several other factors might also have contributed to this undesirable situation. There was a flash fire accident at Oronite's Jurong Island (Singapore) plant in March 2005 which led to the plant operating at reduced levels for several weeks. Oronite's Belle Chasse (Louisiana) plant was scheduled to undergo a 2.5-week turnaround in October 2005, but Hurricane Katrina put this plant out of action in late August 2005. In the aftermath of this incident, Sullivan [1] remarked, "Now it is back on track, but Oronite has taken the position that back to normal is not good enough. The company has vowed to become a more reliable supplier and has begun taking a number of steps toward that end – increasing inventories, establishing supply-chain redundancies and keeping open a plant that was scheduled to be shut."

The Oronite example above illustrates the importance of supply chain management (SCM) and its challenging nature as supply chains (SCs) span across continents, involve numerous entities, and contend with various uncertainties. In this paper, we aim to develop a simulation model of the lube additive SC for decision support, capturing a whole range of knowns and possible unknowns in the SC operation. The simulation model will enable a SC manager to assess the impact of uncertainties on business performance and design suitable policies to define tradeoffs.

Decision-making in the lube additive SC operation is distributed across the various departments in the enterprise and external entities such as customers, logistics providers, and suppliers. Each department has its own goals and tasks. For example, the operations department makes products and the storage department manages inventory. These departments communicate with one another to get the information necessary to perform their tasks or to call for certain actions. The integrated, overall SC performance and its dynamics emerge from the combined effects of the SC entities.

There is limited literature on integrated modeling of a multi-site lube additive SC. The general SC research has focused mainly on the discrete industries [2]. Mathematical programming models have been proposed for managing the multi-site SC. Timpe and Kallrath [3] presented a general mixed-integer linear programming (MILP) model for planning of production, distribution, and marketing for the multi-site SC. Dondo et al. [4] focused on the management of logistic activities in the multi-site SC, proposing a MILP model for the problem of multiple-vehicle pickup and delivery with time windows. Mathematical programming approaches generally do not work as well for larger scale, stochastic problems as the search space may grow exponentially and computational time becomes an issue.

Moon et al. [5] presented an integrated process planning and scheduling model for the multi-plant SC, which includes operations sequencing, machine selection, and operations scheduling. Their proposed solution strategy employs a genetic algorithm-based heuristic approach. Julka et al. [6] proposed an agent-based simulation model of

the refinery SC, where each SC entity is modeled as an agent. Agent-based approach is a natural way to model a system like a SC, where actions of individual components determine system-level behaviors. In this paper, the lube additive SC is modeled as an agent-based model in Jadex [7] using the belief-desire-intention (BDI) formalism [8].

The rest of the paper is organized as follows. The multi-site lube additive SC operation is described in Section 2. The proposed agent-based model is discussed in Section 3, followed by a case study in Section 4. Finally, some concluding remarks are given in Section 5.

2 Multi-Site Lube Additive Supply Chain

Fig. 1 shows a schematic of the lube additive SC, which comprises customers, the lube additive enterprise, and raw material suppliers. The lube additive enterprise comprises a central sales department located at the headquarter and a number of production plants in different geographical locations [9]. Each plant has its own functional departments performing the different SC functions, i.e. commercial, scheduling, operations, packaging, storage, procurement, and logistics. Materials flow from suppliers to the plants and from the plants to customers (solid arrows in Fig. 1) as controlled by the information exchanges (dotted arrows). Customers place their orders with the central sales department, who then communicates with the different plants before deciding to accept and assign a particular order to one of the plants, or to reject it. The plants communicate with suppliers to procure raw materials. Three different cycles of activities constitute the SC operation: enterprise-level coordination, plant operation, and inventory management.

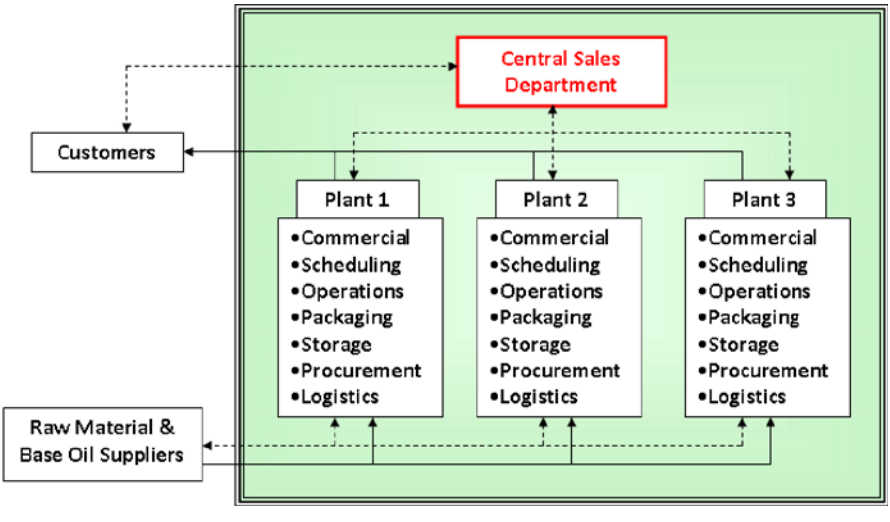


Fig. 1. Multi-site lube additive supply chain

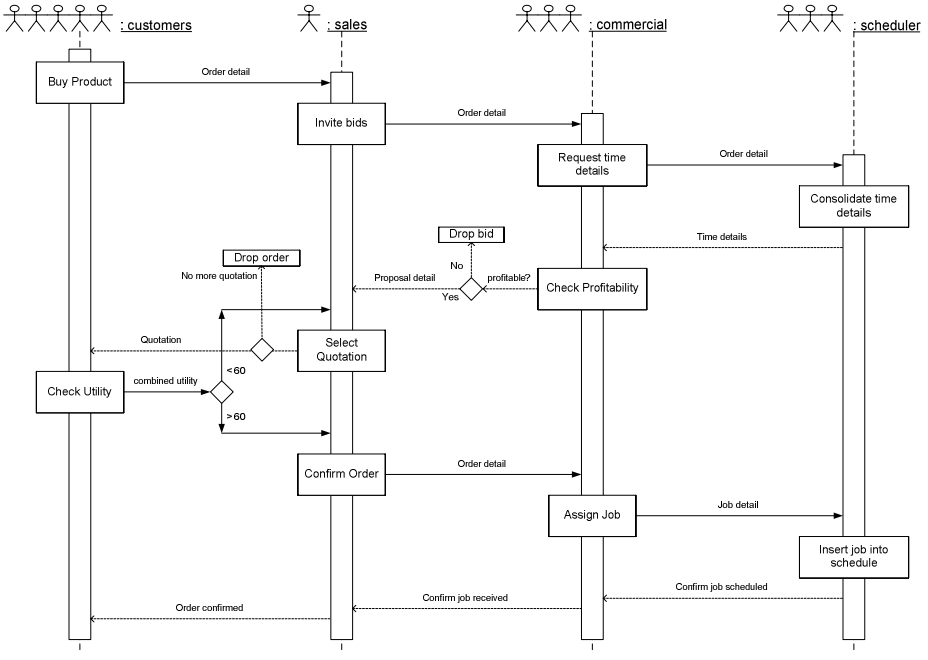


Fig. 2. Sequence diagram of enterprise-level coordination

The SC operation can be represented clearly and conveniently using sequence diagrams. Fig. 2 shows the sequence diagram of the enterprise-level coordination cycle. The diagram shows the sequence of activities (or tasks) performed by the different entities and the interactions involved within one cycle. Each entity has a vertical thread. The entity's name is listed at the top and the human-stick figure illustrates if the entity consists of just one or multiple instances. A vertical dotted line is the time axis on which a task, represented by a rectangle, is placed. A solid arrow represents a message from one entity to another, while a dotted arrow represents a reply. The vertical position of a task shows its sequence relative to other tasks of any entity; the earlier task is placed higher on the time axis. Note that the actual duration of a task or between tasks is not represented in this diagram, only the relative sequence is. A vertical bar shows when an entity is active within the cycle. It starts when the entity first receives a message or performs its first task, and ends after it sends its last message or performs the last task in its thread.

As shown in Fig. 2, the enterprise-level coordination cycle involves multiple customers, one central sales department, and the different plants, specifically their commercial and scheduling departments. The cycle starts with a customer placing an order with sales, who then forwards the order detail to the commercial department of the different plants to invite bids. After consulting scheduler, commercial replies to sales with a proposal if it decides that the order is profitable. The proposal consists of a

price and a delivery date for the order. Following a certain policy, sales works the proposals from the different plants into quotations for negotiation with the customer. If after negotiation the customer accepts the price and delivery date, sales confirm the order and assign it to the respective commercial department. Scheduler then inserts the job into its schedule based on a scheduling policy. The cycle is repeated for each new customer order.

The other two cycles are at the plant-level. Fig. 3 shows the sequence diagram of the plant operation cycle. Scheduler drives the cycle by releasing a job from its schedule, after checking raw material availability with storage, to be processed by operations and packaging. The finished products are sent to customers through logistics and 3PLs (third party logistics providers). The next job in the schedule follows and the cycle is repeated. The sequence diagram of the inventory management cycle is given in Fig. 4. This cycle aims to ensure raw material availability for processing jobs. The procurement policy defines the trigger for procurement, e.g. in the reorder point policy, procurement is triggered when raw material inventory falls to or below the reorder point. When triggered, storage informs procurement of the amount and type of raw material to buy from the supplier. The material is delivered to storage through logistics and 3PLs. The three cycles are interrelated: Job schedule of scheduler connects enterprise-level coordination and plant operation and raw material inventory of storage connects enterprise-level coordination and inventory management.

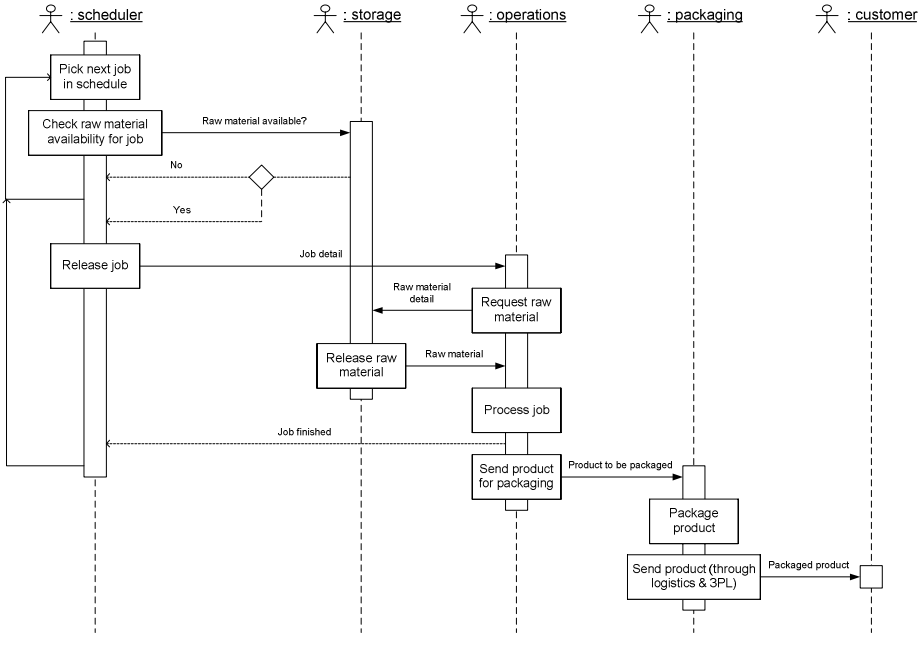


Fig. 3. Sequence diagram of plant operation

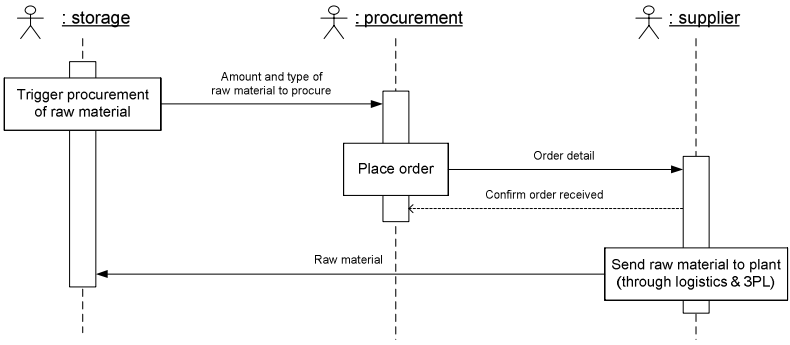


Fig. 4. Sequence diagram of inventory management

2.1 Decisions in Managing the Lube Additive Supply Chain

From the above description, it is clear that managing the supply chain involves decision-making by the different entities. In enterprise-level coordination, scheduler decides how to slot a potential job into the existing schedule and what completion date it can commit. Based on this, commercial decides the details of the proposal to be submitted to sales. Sales then decides what quotations to give the customer and negotiates accordingly. Once agreement is reached, sales decides which plant to assign the job to and scheduler of the corresponding plant updates its schedule. In plant operation, scheduler decides which job to process. It can follow the existing schedule or it may need to reschedule, e.g. if raw material for a particular job is not available. In inventory management, storage and procurement decides when, how much and which raw material to buy. Besides these routine decisions, there are decisions to be made in exceptional circumstances, e.g. how to deal with disruptions such as plant disruption, supply disruption, etc.

The nature of these decisions is mostly decentralized as they are made by different entities. The agent paradigm fits these characteristic well and the decision support problems can be formulated naturally. For example, each plant “bids” for jobs, or plants “collaborate” to deal with situations when one is affected by a disruption. Hence, decision support for managing the SC can be provided by an agent-based model of the enterprise and the SC.

3 Agent-Based Model of the Lube Additive Manufacturing Enterprise

The multi-site lube additive SC described above has been modeled as an agent-based model in Jadex [7] using the BDI formalism [8]. The SC entities in Figs. 2-4, which include external entities (customer, supplier), headquarter entity (sales), and local plant entities (commercial, scheduler, operations, packaging, storage, procurement), are each modeled as an agent. In the current version of the model, the logistics department and the 3PL are not explicitly modeled; delivery of raw materials is modeled as a time lag between the purchase time and their arrival at the plant. In addition to the agents representing real SC entities, the model has two auxiliary agents to facilitate simulation:

manager and time. The manager agent creates all other agents and the graphical user interface. The time agent manages the simulation clock. It starts the simulation, informs all agents of the current time tick, advances the clock when all events within the tick have been completed, and ends the simulation.

In the BDI formalism, agent rationality is modeled through the three mental attitudes of belief, desire, and intention. Beliefs are facts and informational attitudes which keep the agent aware of its internal states and external changes. Agents communicate through messages and update their beliefs as they receive information. Desires are synonymous to goals, which represent the ideal end states the agent is working towards. Intentions are synonymous to plans, which are the courses of actions taken by the agent to reach its goals. The tasks in Figs. 2-4 are captured through plans. A plan may consist of one task or a sequence of tasks. A plan can be triggered by an agent's own goal or by a message from another agent. For example, in Fig. 4 the task "Trigger procurement of raw material" of storage is triggered by its goal of maintaining raw material inventory, whereas the task "Place order" of procurement is triggered by a message from storage.

In Jadex, capabilities are used for the modularization of agents. Capability definition files can be written to encapsulate sets of beliefs, goals, and plans, and contain ready-to-use functionalities. There are several predefined capabilities in Jadex. Of particular interest in the development of the simulation model is the Interaction Protocols Capability which facilitates the usage of some of the predefined FIPA (Foundation for Intelligent Physical Agents) interaction protocols. Two types of interaction protocols are utilized in the simulation model, i.e. the Iterated Contract Net Protocol (ICNP) and the Request Interaction Protocol (RP).

ICNP allows for the negotiations between one initiator and a number of participant agents. More than one negotiation round may be performed, with the purpose to delegate a task to the participant agents and let them execute the task on the initiator's behalf. The interaction between customer and sales prior to the confirmation of the customer order involves the mutual negotiation of quotations devised by sales (Fig. 2). ICNP exhibits features that assist the implementation of this interaction between customer and sales. As such, customer assumes the role of the initiator whereas sales will act as the participant.

The ICNP interaction begins with the customer initiating a "call for proposals" (cfp) by sending the product order details that it desires. Sales will then consolidate job proposals from the plants' commercial departments and formulate a set of profitable quotations, based on its goal to maximise the corporation's profit. The quotation which best fits the customer's due date requirement is quoted at the highest price amongst the available job proposals to the customer. The customer then evaluates the utility value of the quotation according to some price and date utility functions. If the quotation meets the utility criteria, with a combined utility value of, for example, 60 or greater, it is deemed acceptable. The customer will, thus, end the negotiation round and task the sales department to commit to this confirmed order details. Otherwise, rounds of negotiation can be carried out to evaluate the remaining quotations that are available until the customer finally accepts one that fits its utility requirement, or until no quotation is left for evaluation. In the latter case, the intended purchase for product will be dropped by the customer.

RP handles the interaction between one initiator and one participant agent in which the initiator wants the participant to perform some actions. The participant can then

decide if it is willing to execute the particular task that is being requested. Unlike ICNP, there is no negotiation round involved. RP is used in the interaction between sales and each plant's commercial department to allow for the rational and autonomous decision making of the department in the bid for a job. Under this protocol, sales will act as the initiator while the commercial department is the participant. Prior to the sales replying to the customer with quotations in the ICNP communication, job proposal will have to come from each commercial department. Thus, sales will send a request for bid to each commercial department to initiate this RP interaction.

The commercial department will then check with its scheduler counterpart for time information on which it will base the cost estimation for the job. Also, the commercial department will forecast the plant's utilization level for that time period and decide on the price to charge. With this, the commercial department can then make an independent decision as to whether it should participate in the bid for the job. If the formulated price is higher than the estimated cost, the job is deemed profitable for the plant and it will reply to sales with its job proposal. Otherwise, it will not participate in the bid.

In this model, ICNP is used only in the interaction between customer and sales. RP is used in many other interactions between the agents as described in Figs. 2-4.

3.1 Policies as Models of Decision Making

The decisions described in Section 2.1 are modeled through policies. Scheduler has a policy for scheduling and estimating completion date of a potential job. Commercial has a policy to set the price for the potential job in its proposal to sales. Sales has a policy to formulate the quotations for negotiation with the customer. It also assigns the order to a selected plant following a job assignment policy. Procurement buys raw material following a procurement policy.

Different policies can be employed for each purpose. In job assignment policy, sales may assign the job to the plant nearest to the customer, or the plant which is the least busy, or the plant with the lowest cost, or a combination of these considerations and others. Procurement can be done on a periodic basis or when a certain inventory level is reached. Scheduling can be done by sequencing based on a certain parameter, e.g. shortest processing time, first-come-first-serve, earliest due date, or more elaborate algorithms.

4 Case Studies

The lube additive enterprise considered here has three make-to-order plants located in Houston, Japan, and Singapore. Different plants may have different operating costs, efficiency, and raw material delivery time. A more efficient plant processes the same job faster than a less efficient one. Locations of plants, suppliers, and customers are represented using (x, y) coordinates on a 2-D plane. Transportation time between two locations is assumed to be linear to their straight line distance. There are three product types, each with five different grades; each grade is produced using a specific recipe from five additives and three base oils. Customer's orders (type, grade, amount, due date range) are randomly generated based on a demand curve. Processing and packaging times are deterministic. Each plant has only one processing line; only one job can be processed at a time.

In this case study, the three scheduling departments employ the PEDD (processing earliest due date) scheduling policy. Each customer order comes with a due date range, from earliest due date (EDD) to latest due date (LDD). PEDD is the earliest date to start processing to meet the order's EDD:

PEDD = EDD - Delivery Time - Packaging Time - Processing Time

PEDD refers to the earliest date that the plant should start processing the job in order for the finished product to reach the customer by the agreed EDD. Under this policy, the new jobs are sequenced such that the jobs with the earlier PEDDs are placed with higher priorities. The procurement policy used is the reorder point policy. Under this policy, the procurement department will raise a purchase order for raw materials when it is informed that the inventory levels have fallen to or below the reorder points, to bring them up to the top-up points. Procurement will keep track of the pending inventories that have been purchased but have not arrived. This is important to avoid any repeat orders for raw materials.

4.1 Case Study 1: Dealing with an Unreliable 3PL

The SC manager is interested in studying the impacts of the unreliable 3PL on the overall performance. This unreliable 3PL has a 70% probability of late raw material delivery, which can take 80% to 90% longer than the normal expected delivery time. A simulation is run for 200 days with a total of 200 processed jobs assigned to the three plants. In this first run, the reorder points used in the procurement policy are [1000 for base oils, 100 for additives]. The results are given in Table 1(a). The total tardiness for 200 jobs from the three plants is 110 days, which is very high and leads to poor customer satisfaction. To study if this can be improved, a second simulation run is done using higher reorder points of [4500 for base oils, 700 for additives]. The results are given in Table 1(b). Fig. 5 shows the inventory of base oils in the three plants, for both cases of low and high reorder points. The horizontal dotted lines indicate the reorder points.

The orders from customers are the same in both cases. From Table 1, we can see that the number of jobs processed by each plant in the two cases is different. This is due to different job assignments, which depend on the bids submitted by each plant, which in turn depend on the existing schedule, which may differ due to raw material availability. Consequently, the total amount of product delivered by each plant is also different in the two cases.

The higher reorder points improve the total tardiness to 43 days, down by 61%. This is because the raw material inventory is kept high (see Fig. 5) such that there is less probability of a job being delayed due to insufficient raw materials. A penalty is imposed for late delivery proportional to the tardiness. The higher reorder points thus also reduce late penalties by 61% to $215K. On the other hand, inventory cost increases by 39% to $1012.5K, as higher inventory is maintained by having higher reorder points. Overall, the profit decreases by 4%. Further experiments can be performed to find more optimal reorder points. This case study demonstrates how the simulation model can be used to study such tradeoffs and provide decision support.

Table 1. Simulation results for unreliable 3PL case study

	(a) Low Reorder Points			
	Houston	Japan	Singapore	Total
Jobs Processed	70	44	86	200
Product Delivered (volume unit)	63943	49212	67435	180590
Tardiness (days)	19	47	44	110
Late Penalties (k$)	95	235	220	550
Inventory Cost (k$)	285.6	245.7	195.9	727.2
Profit (million $)	5.10	3.22	6.02	14.34

	(b) High Reorder Points			
	Houston	Japan	Singapore	Total
Jobs Processed	73	58	69	200
Product Delivered (volume unit)	67554	55311	55472	178337
Tardiness (days)	9	1	33	43
Late Penalties (k$)	45	5	165	215
Inventory Cost (k$)	398.2	340.7	273.6	1012.5
Profit (million $)	5.30	4.01	4.52	13.82

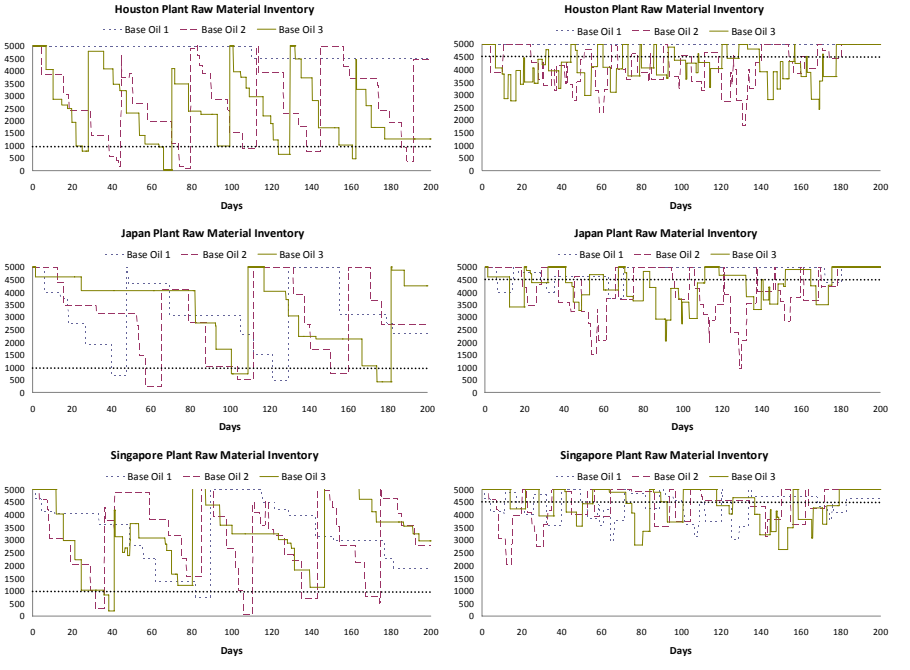


Fig. 5. Raw material inventory of the three plants for low reorder point (left) and high reorder point (right) for unreliable 3PL case study

4.2 Case Study 2: Effect of Production Scheduling Policy

In this case study, the SC manager notices that many customer orders are delivered late. He consults with the schedulers and finds that they are using the PEDD scheduling policy. Seeing that this policy, when inserting a new job to a slot in the schedule, does not consider the due date of the jobs in the slots behind it, he proposes a new scheduling policy. This policy is similar to PEDD with one modification: scheduler will check to ensure that the insertion of the new job into the schedule will not result in any of the jobs behind it becoming late. Otherwise, the new job will be moved to the next slot, until no jobs behind it become late with its insertion. Thus a new job insertion should not affect the completion times of previously committed jobs such that they become late. These two policies – PEDD and PEDD-with-Late-Jobs-Consideration – are examined through simulation and the results are shown in Table 2. The new policy significantly improves the total tardiness from 73 days to 9 days (down by 88%) with comparable profit. This shows that a small modification to a policy could have a big impact. This case study demonstrates how the simulation model provides decision support for policy evaluation.

Table 2. Simulation results for new scheduling policy case study

	(a) PEDD			
	Houston	Japan	Singapore	Total
Jobs Processed	74	54	72	200
Product Delivered (volume unit)	69706	49819	59019	178,544
Tardiness (days)	22	22	29	73
Late Penalties (k$)	110	110	150	370
Profit (million $)	5.48	3.36	4.65	13.49

	(b) PEDD-with-Late-Jobs-Consideration			
	Houston	Japan	Singapore	Total
Jobs Processed	64	55	81	200
Product Delivered (volume unit)	64810	52107	61110	178,027
Tardiness (days)	5	2	2	9
Late Penalties (k$)	30	10	10	50
Profit (million $)	4.84	3.27	5.04	13.15

5 Concluding Remarks

The complex dynamics in a multi-site lube additive manufacturing enterprise necessitate the development of a simulation model to support SCM. The agent-based model presented in this paper adequately captures the integrated SC operation. The modular structure of the model makes it easy to extend, for example to include a new plant. The model enables simulation experiments to analyze policies, optimize parameters, evaluate impacts of a disruption, devise disruption management strategies, provide decision support at the strategic, tactical, and operational levels. The model can be extended in a number of ways. It can be made more scalable by employing parallel

computing where the agents are distributed across multiple processors. Explicit modeling of logistics and 3PL agents will add to the model's depth and scope. Adaptive optimization studies can be performed by giving each agent the capability to adapt and change its parameters to maximize its utility as the simulation progresses. The same modeling approach can also be extended to supply chains of other products.

Acknowledgments. We are indebted to Dr. Jean-Luc Herbeaux and Ms. Bonnie Tully of Evonik (Degussa) South-East Asia for sharing their valuable insights and knowledge on the multi-site lube additive supply chain operation.

References

1. Sullivan, T.: Special Report: Road to Recovery – Oronite Strives to Regain Customer Confidence. Lube Report 6(25), June 21 (2006)
2. Srinivasan, R., Karimi, I.A., Vania, A.G.: Business decision making in the chemical industry: PSE opportunities. In: Marquardt, W., Pantelides, C. (eds.) Computer-aided Chemical Engineering, vol. 21, pp. 107–117. Elsevier, Amsterdam (2006)
3. Timpe, C.H., Kallrath, J.: Optimal planning in large multi-site production networks. Eur. J. Op. Res. 126(2), 422–435 (2000)
4. Dondo, R., Mendez, C.A., Cerda, J.: Optimal management of logistic activities in multi-site environments. Comput. Chem. Eng. 32(11), 2547–2569 (2008)
5. Moon, C., Kim, J., Hur, S.: Integrated process planning and scheduling with minimizing total tardiness in multi-plants supply chain. Comput. Ind. Eng. 43(1-2), 331–349 (2002)
6. Julka, N., Karimi, I., Srinivasan, R.: Agent-based supply chain management – 2: a refinery application. Comput. Chem. Eng. 26(12), 1771–1781 (2002)
7. Pokahr, A., Braubach, L., Lamersdorf, W.: Jadex: a BDI reasoning engine. In: Multi-Agent Programming, vol. 15, pp. 149–174. Springer, US (2005)
8. Rao, A.S., Georgeff, M.P.: BDI agents: from theory to practice. In: Proceedings of the First International Conference on Multiagent Systems, San Francisco, CA (1995)
9. Zhang, H., Wong, C.W.K., Adhitya, A., Srinivasan, R.: Agent-based Simulation of a Specialty Chemicals Supply Chain. In: Zhang, H., Wong, C.W.K., Adhitya, A., Srinivasan, R. (eds.) International Conference on Infrastructure Systems: Building Networks for a Brighter Future, Rotterdam, The Netherlands, November 10-12 (2008)

Embodied Organisations in MAS Environments

Michele Piunti[1], Alessandro Ricci[1], Olivier Boissier[2], and Jomi F. Hübner[2,3]

[1] Università degli studi di Bologna - DEIS, Bologna, Italy
{michele.piunti,a.ricci}@unibo.it
[2] Ecole Nationale Superieure des Mines - G2I St-Etienne, France
{boissier,hubner}@emse.fr
[3] Federal University of Santa Catarina - DAS, Florianópolis, Brazil
jomi@das.ufsc.br

Abstract. Agents and Artifacts model extended with organisation promotes artifact based environments aimed at supporting multiagent coordination and goal oriented interactions and communication. Nevertheless, the use of artifacts for organisational purposes constrains agents to be aware of complex structures described in an organisational specification: an organisational specification: for instance, agents have to understand and be able to manipulate low level primitives which may be not proper of an application domain. To ease this requirement, we propose "organisational embodiment rules" as a programmable layer for building embodied organisational artifacts (EOA) through their binding to environment artifacts. EOAs are aimed at transparently interceding with the organisational structures, and at enabling possibly organisation-unaware agents to seamlessly play in organisations with no need to deal with low level mechanisms of an organisational specification. We propose a formal description along with examples enlightening benefits of the proposed approach with respect to related ones.

1 Introduction

A visitor entering in an organized environment has no need to directly communicate with the organisation which is operating in such context, nor he/she needs to exhaustively know how such organisation is structured. Indeed complex environments where human beings live are often instrumented with media, resources, services and embodied objects aimed at assisting human activities and letting users unaware of participating to complex organisational patterns. This is why, for instance, visitors entering in an hospital through a "visit" door have no way to directly communicate with the hospital as an organisational entity, nor need they a complete knowledge of the multiple organisational schemes that are in action behind it. However, hospitals typically deploy several facilities in order to promote specific patterns of cooperation, to which visitors unawarely participate, i.e. by using those environment artifacts as support mechanisms which can be straightforwardly afforded.

Following this suggestion, the contribution of this work aims at bridging the gap between actual agents and organisational programming models in Multi Agent Systems (MAS). Being in an instrumented work environments means, for computational agents as well as for visitors in the hospital, to not be forced to explicitly interact with the organisation, nor to have an explicit representation of the organisation in mind in order to

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 115–127, 2009.

exploit its services. The proposed approach assumes agents' worlds to be instrumented by Embodied Organisational Artifacts (EOAs) supporting either interaction with the environment resources, either the management of the organisation life cycle. Our major concern is to conceive, on such basis, rules to embody organisational artifacts into the environment artifacts. In particular we here provide a design model according to which heterogeneous agents can concurrently operate in the same work environment being coordinated and controlled by situated artifacts aimed at supporting such social activities. After having discussed related works in Section 2, the A&A computational model is formally introduced in Subsection 3.1 while Subsection 3.2 synthesises *M*OISE organisational specification and its artifact based implementation ORA4MAS. The programming model for instrumenting work environments with environment and organisational artifacts is introduced in Section 4 and further described in Section 5 with an example showing system dynamics.

2 Background

The complex requirements of wired and distributed software systems, as well as the increasing computational power of hardware platforms are originating a growing interest towards organisation oriented programming (see, among others, [6]). Recent developments in MAS area originated many proposals in this direction, which are straightforwardly rooted in typical perspectives of agent oriented modeling. As noticed in [7], a recurrent element in proposed approaches is the presence of middle-ware components (organisational proxies) that actually mediate interactions between agents participating the organisation and the infrastructures providing organisational services. On these basis, a series of modeling drawbacks rise when fine grained and cognitive interactions between agents and organisational middleware are of concern. A first major issue is the mismatch at the abstraction level between the entities playing the system. Whereas the organisation is conceived as a series of services at a slightly "abstract" level, i.e. defined in terms of norms, roles, global goals, etc., agents are built upon mentalistic notions as beliefs, desires, intentions (assuming that BDI-like agents are in place). This places the problem to bridge the conceptual gap between the elements and the resources available at the organisational level and the notions and the constructs adopted at the agent level. Moreover, the designers (and the agents) have typically to deal with a twofold interaction medium composed by a *virtual* organisation (where agents adopt roles, commit missions, coordinate themselves for fulfilling joint activities and workflows) and a physical, *embodied* environment[1] (where agents perform actions, perceive events, communicate, move, access shared resources etc.). Although their concerns are similar, their reification usually leads to different approaches: dedicated layer in the case of organisations, proper entities exploitable by agents in the case of environment. Finally, middleware based organisations bias an overwhelming power over agents which are simply participants, hence placing the issue to play an active role in managing the system, i.e. by creating, modifying, adapting the organisation on the need.

[1] *Embodied* is here an abuse of notation. The term has to be intended as *reified entity of agents computational world*, more than in its cybernetic notion of having a physical body.

Recent approaches have been addressed at bridging the conceptual gap between organisations, environments, agents by introducing organisation situated in MAS environments. Based on programming languages used to describe environments and normative infrastructures, Okuyama et al. proposed a model for situated organisations instrumenting concrete environments where social interactions are of concern [9]. The proposal is to distribute *normative objects* as reactive entities readable by agents working in *normative places*. The approach is assumed to improve emergent dynamics governed by specific norms addressed at controlling agents behavior. In fact, normative objects are supposed to indicate obligations, prohibitions, rights and are indeed affordable pieces of information that agents can get and reason about with no previous knowledge.

The works on Electronic institutions (EI) envisaged particular organisational proxies mediating between agents and organisation. Governors entities are assumed to rule and enable the interactions in respect of communication protocols and norms [2]. Situated Electronic Institutions (SEI) was recently proposed as an extension aimed at interceding between environments (real world) and EI. In this case, special governors, namely modelers, allow to bridge environmental structures to the EI by instrumenting environments with embodied devices controlled by the institutions. Participating agents can, in this case, perform individual actions and interactions (either non message based) while being monitored by the institution. Besides, staff agents are assumed to operate on concrete world devices, i.e. judging norm violations.

The notion of observability of environmental states is also a pivotal one in the formal model proposed by Dastani et al. [4] where agents behavior can be regulated by regimenting and sanctioning rules which are triggered by the monitoring process of the overall environment. The organisational module is here assumed to contain norms defined on the basis of "count-as" and "sanctioning" rules, specified in terms of transitions regulating the effects of the actions performed by agents in their environment.

As detailed in the next sections, the model presented here mainly differs form the described ones for being natively conceived in terms of agents and artifacts (A&A). In our proposal the overall organisational infrastructure is modeled in terms of distributed artifacts, which are reactive components to be suitably exploited by agents for their *epistemic* (artifacts have an observable state) and *pragmatic* (artifacts have triggerable operations) purposes. A&A interactions obey to unambiguous rules defined at a platform level, and it is modeled through agent native capabilities of action and perception. Whereas artifact based environments works on a dedicated platform (i.e. CArtAgO), agents can share the same organisational services simply by integrating the repertoire of action and perception needed to join a workspace and work with artifacts. In addition, the computational model of artifacts is conceived on a rich set of relevant events to be perceived by agents. More than in terms of observable states, the proposed approach allows to deal with relevant changes occurring in the environments in terms of artifact *events* which can be distributed across several workspaces and directly addressed to agent's reasoning processes in order to be handled. Basically this enables designers (and agents) to recognize system dynamics at a finer granularity, effectively improving situatedness of the system. Finally, the functional link between artifacts allows a comprehensive approach of the organisation, which is unambiguously divided in two modules placed at two distinct conceptual levels: an organisational level containing a

detailed organisational specification (including norms, roles, groups specification) and an embodied level, containing those environment artifacts to be monitored and controlled by the organisation itself.

3 Agents and Artifact Systems

Before describing artifact based organisations, this section resume A&A and CArtAgO respectively as the conceptual model and the related computational platform at the basis of the proposed architecture.

3.1 Artifact Based Work Environments and CArtAgO

For the sake of simplicity, the CArtAgO computational model at the basis of artifact based work environments presented here abstracts away from insights concerning agent execution model, as well as a simplified model of artifacts is sketched (a more detailed description is in [12]). We here adopt a formalism which is compatible with models of agent which have been integrated in CArtAgO (i.e. *Jason/AgentSpeak*, *2APL*, *Jadex*, [10]). Some conventions are adopted, as the following meta-variables: *instate* ranging over artifact state, τ over artifact templates, and σ over workspaces. Meta-variables for the unique identifiers of agents and artifacts are denoted respectively as γ and α. Given meta-variable x, a set of elements of kind x is denoted as $\overline{x}$. Besides, meta-variables that have no definition (like *instate*, *manual*, *mstate*, or any identifier) correspond to constructs whose structure is not prescribed by a system specification, i.e. their details depend on the actual integration with agent platforms.

A&A MAS. As depicted in Fig. 1, an agent and artifact system ($AA - MAS$) is a composition of agents ($\overline{Agent}$) playing in environments structured in workspaces ($\overline{Workspace}$). An agent is represented by its identifier (*AgentName*) and its computational model (*AgentProg*). Agents are assumed to join and work in multiple workspaces at a time: joining a workspace, an *AgentBody* is created as interface between the mind part – which depends on the specific agent model/architecture adopted– and the environment, and it contains sensors and effectors allowing interaction with artifacts belonging to that workspace. A workspace is thus defined by an identifier (σ), a set of artifacts ($\overline{Artifact}$) and a set of agent bodies ($\overline{AgentBody}$).

Agents. Agents are those autonomous, *pro-active* entities designed in order to achieve some kind of goal or task. Agents are also *reactive*, i.e. they are meant to continuously perceive input from the environment and accordingly react. In this view, agent's activities are based on *perception* and *action*. According to Fig. 1, an agent is defined at the system level by its name (*AgentName*) and its system identifier γ. More than on agent execution and programming model we are here interested in defining au unambiguous interaction model defining agent-artifact interaction. In CArtAgO it is based on a series of basic actions to be included in agent's repertoire. The tenet of this repertoire is based on the activities of *action* (aimed at changing the world) and *perception* (aimed at perceiving the world). Indeed, use, sense and observeProperty are the pivotal actions which agent have in repertoire to require artifact's operations and observe artifact observable state. By executing focus or stopFocus, agents can initiate/terminate

$AA-MAS$	$::= \langle \overline{Agent}, \overline{Workspace} \rangle$	A&A System
$Agent$	$::= \langle agentName, agentProg \rangle$	Agent Entity
$Workspace$	$::= \langle \sigma, \overline{Artifact}, \overline{AgentBody} \rangle$	Working space element
$Artifact$	$::= \langle artifactName, artifactProg \rangle$	Artifact Entity
$artifactName$	$::= \langle \alpha, \tau \rangle.$	Artifact
$artifactProg$	$::= \langle \overline{uic}, \overline{a-obs-prop}, manual, instate \rangle$	Artifact program
uic	$::= \langle [guard] opName \rangle$	Usage interface control
$a-obs-prop$	$::= \langle artifactName, pname \mapsto pvalue \rangle$	Artif. Observable property
$a-ev$	$::= \langle a-prop-ev \rangle \mid \langle a-op-ev \rangle$	Artifact Event
$a-prop-ev$	$::= \langle \alpha : pname(evtValue : evtType) \rangle$	Artif. Property(-related) event
$a-op-ev$	$::= \langle \alpha : opName(evValue : evType) \rangle$	Artifact Operation(-related) event

Fig. 1. Definition of a Multi Agent System in the A&A model and CArtAgO

the long term activity aimed at perceiving artifact relevant changes in terms of events, without directly operating over it. Other actions used by agents (i.e. to join and leave workspaces, to create and dispose artifacts, etc.) are not detailed here.

Workspaces and Artifacts. Besides agents, an A&A system is modeled in terms of *workspaces*, representing virtual locations containing sets of first-class entities called *artifacts*. Artifacts are the basic building block of MAS environments, they represent reactive and automatic resources that agents can share and use to ease their work. In this view, artifacts are non-autonomous, *function-oriented* entities, designed to encapsulate functionalities that agents can exploit, through an interface of available operations, to achieve their goals. The set of artifacts in a workspace is dynamic: agents have basic actions to dynamically enter/exit workspaces, and to create/dispose artifacts etc. An artifact is issued from a library of templates (τ) and is defined by its name *ArtifactName* and by a system identifier (α) controlled by the environment platform. The *ArtifactProg* element includes artifact execution model which is defined from the artifact usage interface ($\overline{uic}$), observable properties ($\overline{prop}$), and finally an inner state (*instate*). The usage interface includes a set of controls, each providing the name of the triggerable operation (*op-name*) and a possible guard function enabling the operation execution (*guard*). Similarly to artifacts used by human, artifact computational model presents a *manual* element as an additional construct inspectable by agents to "learn" their functioning. The manual description, as well as the inner state and the artifact operation processes denoting the internal execution model are not detailed here for simplicity.

Artifact computational model assumes observable states, events and signals as the pivotal constructs to acknowledge agents about world changes. On the one side, the execution of artifact's operations – which occur asynchronously to agent processes – can lead to the generation of *observable events* that the agents can perceive. Besides events, each artifact can expose an observable state, in terms of one or multiple *observable properties* whose value can be dynamically perceived by agents, without triggering operations or processes over the artifact. As presented in Fig. 1, there are two types of perceivable facts: *(i) a-obs-prop*, indicating observable properties, each specifying artifact identifier (*ArtifactName*), property name (*pname*), and property value (*pvalue*) and *(ii) a-ev*, indicating events generated by the artifacts that the agent is scrutinizing, called artifact events. In more details, an *a-ev* can be related to events coming from

artifact observable properties (*a-prop-ev*), each specifying artifact identifier (α), event value (*evtValue*), and event type (*evtType*). Besides, *a-ev* can also indicate events related to the execution of artifact operations (*a-op-ev*), namely events signalled by operation processes, each specifying artifact identifier (α), the executed operation (*opName*), and information about the outcome/result of the operation (*resValue* and *resType*). Whereas A&A provides a conceptual design model, CArtAgO [1] puts in practice the enabling technology and provides a programming platform to create and manage artifact based working environments to which heterogeneous agents can join [10].

3.2 Artifact Based Organisations

Either the A&A conceptual model and the CArtAgO programming platform presented in the above section allow the realisation of open environments based on artifacts and workspaces where heterogeneous agents can play and cooperate in joint activities. The envisaged model lacks of explicit organisational infrastructures, i.e. the presence of normative and institutional elements allowing to tackle directives at the system level as ruling and governing agents behavior and social dynamics. To bridge this gap, ORA4MAS has been recently proposed as an artifact based infrastructure, based on CArtAgO, where organisational elements are modeled as artifacts, thus as first class entities of the system [7]. The ORA4MAS artifacts are shaped on the basis of *M*OISE organisation modeling language [8]. *M*OISE is adopted to introduce normative elements and specific cooperation patterns that allow to explicitly decompose the specification of an organisation into structural, functional, and deontic dimensions. The structural dimension specifies the *roles*, *groups*, and *links* of the organisation. The definition of roles states that when an agent decides to play some role in a group, it is accepting some behavioral constraints related to this role. The functional dimension specifies how the *global collective goals* should be achieved, i.e. how these goals are decomposed (in global *plans*), grouped in coherent sets (i.e. *missions*) to be distributed to the agents. The decomposition results in a goal-tree, called *scheme*, where the leaves-goals can be individually achieved by agents. The deontic dimension is added in order to bind the structural dimension with the functional one by the specification of norms (i.e. *permissions* and *obligations* to be related to roles and missions).

In ORA4MAS, each dimension of *M*OISE is managed by a specialized type of artifact. Thus, three types of organisational artifacts (OAs henceforth) are conceived: group, scheme, and normative artifacts. Each of them is automatically initialized on the basis of the specification language –based on *M*OISE– contained in *Norm-Org-Exp* statements (Fig. 2) Among ORA4MAS artifacts, the `groupBoard` artifact maintains the state of an instance of group and provide operations related to this group. `adoptRole(`ρ`)` is used by an agent to adopt a new role ρ in the group and `leaveRole(`ρ`)` is used to give up the role. As observable properties, the group artifact shows the current status of the collective plan specified in the scheme in terms of agents and their actual roles in the group. The `schemeBoard` artifact provides operations related to the execution of an instance of a scheme: the operation `commitMission(`m`)` have to be used by an agent to commit to the mission m; `leaveMission(`m`)` to decommit mission m and `setGoalAchieved(`g`)` to notify the achievement of a goal g. As observable properties, it shows the list of agents engaged with the scheme, their missions in the scheme and the state of corresponding

goals. Finally, the `normBoard` artifact provides no *uic* operations but a series of observable properties which are inspectable to know the actual normative state of the organisation. Linked to others OAs, the `normBoard` is conceived to take trace of the current state of the organisation: since changes notified by other OAs, it automatically detects mismatches indicating non-compliance to the norms issued by the organisation. The current state of the norms vis-a-vis the members of the organisation are shown as observable properties so that special agents (i.e. *staff* agents) are able to decide about possible sanctions as specified in organisational specification. For simplicity, we here abstract away from the organisational specification as well as from the description of the effects elicited by OAs operations. A detailed description of ORA4MAS as well as A&A modeling and CArtAgO can be found in [11,7].

4 Situating Organisations in Agents Work Environments

As far as the ORA4MAS model is conceived, in order to exploit the organisation agents need to know OA's functioning before their use, and they need additional capabilities to bring about organisational and normative notions (i.e. role, missions, schemes, obligations etc.) which are not native constructs of their architectures. Similarly to the approaches described in Section 2, to exploit organisational facilities agents need to perform a set of additional activities which are not concerned with the fulfilment of their purposes but that are needed to explicitly notify the organisation about their activities. For instance, an agent who wants to adopt a role needs to explicitly use an `adoptRole` operation upon a particular OA, as well as an agent who achieves a goal needs to explicitly notify it with a `setGoalAchieved` operation. This section shows how such weakness can be effectively bridged by exploiting an additional layer of environment artifacts (EA) situated in the work environment which are aimed at mediating between agents and organisational infrastructures.

4.1 Embodied Organisational Artifacts (EOA)

Taking inspiration from complex organisation in which human beings operate, the proposed environment model is instrumented with embodied resources aimed at assisting agent activities at their micro level. In so doing, an artifact based organisation for an open MAS is assumed to be composed by an heterogeneous set of *Embodied Organisational Artifacts* (EOAs) which can be of two types: EAs and OAs. (*i*) Environmental Artifacts (EAs) are a special kind of embodied resources placed in the working environment which are directly deployed and controlled by the organisation with the aim to rule and enable agent activities. (*ii*) Organisational Artifacts (OAs) embedding general organisational services, as the one discussed in Subsection 3.2.

Whereas OAs provide mechanisms related to the organisational control, EAs embed suitable facilities allowing to situate such organisational control in a work environment: by ruling agents behavior EAs achieve coordination and prevent deviation from equilibrium and undesirable states. By providing embodied organisational resources to be exploited by agents in a transparent fashion, EAs are conceived to enable agents in achieving their individual goals. Besides, EA can be used to mediate between organisation and environment, so to give rise to functional, unaware, collective phenomena

which can be fully controlled by the organisation. On these basis, two kind of agents are envisaged to dwell the work environment: *Staff agents* are explicitly designed to operate upon the organisation, hence they can directly use OAs operations and perceive OAs observable states. As an example, in an hospital scenario, medical and paramedical staff are likely aware about the organisational schemes, as they have capabilities to directly operate upon the organisation or to intentionally exploit its services. On the other side, *unaware agents* are not able to use OA operations, due to a lack in their model or because, for a design choice, they have no access rights. In a hospital scenario, for instance, patients and visitors have no knowledge about the organisational schemes and thus they are likely unable to directly interact with the organisation.

4.2 Relating Organisations and Environments

Deploying environmental facilities aimed at aiding unaware agents tasks is thus established as a functional (programmable) relation between EA based facilities and the OA. To define this relation an abstraction mechanism provided by "count-as" and "enacting" relations is adopted. A relation "count-as" is a special relation between the effects of an action performed by an active entity in a specific context (i.e., an agent upon an artifact) and those special effects produced by that action which can be addressed to the institutional dimension of the system. Institutional actions, i.e., actions performed in the normative context of an organisation, can be considered as particular actions endowed with special effect of "count-as". In this view, a normal action of an agent "count-as" a conventional or an institutional action once it is situated in a particular institutional context [13]. Thus, a given action performed by an agent acquires an additional *effect* ("count-as" empowerment), due to the fact that the organisation recognizes the ongoing activity and ascribe to it a normative outcome. Hence, the "count-as" effect is assumed as a *vehicle* to route environmental facts to institutional ones. By adopting "count-as" mechanisms, EAs are supposed to intercede between agents and organisation. In fact, using or attempting to use an EA operation produces an event that, from the point of view of the organisation in OAs, may "count-as" an institutional fact (and holds to the same effects provided by operations like role adoption, mission commitment etc.). Besides, using an EA operation may produce some undesired effects in environment that "count-as" a norm violation.

Besides count-as, a second effect can be considered, that is an enactment effect dealing with regimentation-enforcement aspects that the organisation may provide to control the system. In this case the organisation is aimed at producing a control effect by enacting changes upon the environment and its embodied artifacts with the aim to promote desirable states of equilibrium. Notice that a normative control that the organisation may want to operate over the environment can be defined also in the normative dimension written in the organisation specification. For example, a norm specifying that a certain door "can be used by medical staff only" can be ensured by user's personal badges or keys and may not require a special enactment rule. In this example, the opening procedure to open the door is the *instrument* that implements that norm. However, norms for visitors like "pay an additional fee" do not need to (or cannot) be forced by any embodied artifact in the same way it can force them to access only authorized area. We are thus considering two main mechanisms to instrument norms, regimentation

$ORG-AA-MAS$	$::= \langle \overline{Agent}, \overline{Workspace}, \overline{Org-Emb-Rule}, \overline{Norm-Org-Exp} \rangle$	A&A Organisational System
$Workspace$	$::= \langle \sigma, \overline{Artifact}, \overline{AgentBody} \rangle$	
$Artifact$	$::= \langle OA \rangle \mid \langle EOA \rangle$	
EA	$::= \langle artifactName, artifactProg \rangle$	Environmental Artifact
OA	$::= \langle artifactName, artifactProg, \overline{Norm-Org-Exp} \rangle$	Organisational Artifact
$Org-Emb-Rule$	$::= \langle count-as \rangle \mid \langle enact \rangle$	Organisational Rules
$count-as$	$::= \langle \alpha : ev \rangle \longrightarrow \langle \alpha : uic \rangle$	Count as Rules
$enact$	$::= \langle \alpha : org-ev \rangle \longrightarrow \langle \alpha : uic \rangle$	Regimentation/Sanctioning Rules
ev	$::= \langle env-ev \rangle \mid \langle org-ev \rangle$	Relevant Event
$env-ev$	$::= \langle a-ev \rangle$	Environment Relevant Event
$org-ev$	$::= \langle a-ev \rangle$	Organisation Relevant Event

Fig. 2. Embodied Organisational Artifacts: the definitions indicate the functional relations between OAs and EAs

and enforcement (this classification is based on the proposal in [5]). *Regimentation* is a mechanism that simply prevents and makes impossible for the agents to perform actions that are forbidden by a norm (i.e., barrier effect). In particular, the organisation regiments some actions in order to preserve important features of the system. For instance, the access to valuable resources as staff working rooms can be physically regimented by the mean of EAs components, i.e. locks or badges readers instrumenting the entering door. *Enforcement* is a mechanism which can be applied after the detection of the violation of some norm. Agents can indeed decide to obey or not the norm according to their local view. From a system point of view, the fulfilment/unfulfillment of the norms should be detected, evaluated as a violation or not, and then judged as worth of sanction/reward or not. For instance, by the mean of an EA which is being used by some malicious agent, the organisation can automatically recognize a norm violation, thus eliciting an enactment providing a sanction/regimentation feedback. It is worth remarking that these two mechanisms allow to balance between constraining pivotal properties of the system without affecting agents' autonomy. This is of particular importance to let the system to self-organize and evolve as in the case of open systems, where heterogeneous agents are supposed to enter and leave with unknown architectures and purposes.

4.3 Organisational Embodiment Rules

This section introduces a rule based mechanism enabling the definition and the regulation of the functional relations between environmental artifacts (EAs) and organisational artifacts (OAs). The proposed mechanism is expressed through the specification of functional linking relating EAs to OAs and viceversa. Fig. 2 slightly changes the description of an A&A system (as it has been proposed in Fig. 1) by introducing an additional organisational layer: an *ORG-AA-MAS* presents an additional series of rules for the embodiment of OAs into EAs (*Org-Emb-Rule*) and a set of normative expressions (*Norm-Org-Exp*). Either EAs and OAs are represented through an artifact identifier (*ArtifactName*) and a proper computational model (*ArtifactProg*). OAs also include *Norm-Org-Exp*, which are a subset of the normative rules specified in *ORG-AA-MAS*. These normative rules are assumed to set the organisation by configuring the related OAs. They can be automatically generated on the basis of the *M*OISE specification, as it has

been introduced in Subsection 3.2. It is worth remarking that, since the translation of *M*OISE specification in terms of *Norm-Org-Exp*, an organisation becomes a normative system, and it is completely defined on the basis of norm expression. For simplicity, the syntax description of *Norm-Org-Exp*, as well as the automatic mechanisms generating them since a *M*OISE specification, will be discussed in future works.

Structures defining *Org-Emb-Rule* in Fig. 2 refer, from the one side, to the environmental events, which in our case are widely grounded to the work environment and coincide with the EA related events (*env-ev*). On the other side, they refer to the organisational events describing those relevant changes happening in the institutional layer in terms of OA events (*org-ev*). Both $env - ev$ and $org - ev$ are then rooted in general artifact relevant events $a - ev$. This allows to manage a series of rules describing the functional relations between EA and OA. In particular, two kinds of *Org-Emb-Rule* are expressed, namely Count-as and Enact rules. **Count-as** rules, state, for an event generated by an artifact α, which are the consequences at the organisational level. *count-as* rules indicate how, since the actions performed by agents, the embodied organisational artifacts automatically detects relevant changes (*ev*), thus relating them to execution of some operation at the organisational level. In other terms, since a relevant event that is generated by an embodied organisational artifact, an operation upon a given OAs is automatically triggered through its *uic* interface. In so doing, either changes occurring in the work environment (possibly operated by agents), either relevant events occurring in the context of the organisation can be further translated in the opportune institutional changes at the OA level, that updates itself accordingly.

Enact rules, state, for each institutional event, which are the control feedback at the environmental level. *enact* rules express how, since the specified *Norm-Org-Exp*, the OAs are assumed to control the EAs. In so doing, organisational events (i.e. role adoption, mission commitment) can be further used to elicit changes in the work environment, i.e. by mean of sanctioning and regimentation mechanisms.

5 An Example: Hospital Scenario

Let us consider the hospital "visit" service introduced in Section 1 as an example of embodied organisation. The organisation assumes that a visitor agent (*va*) opening the "visit door" is entering the room and playing the role "patient", which purpose is to go to the "visit desk" to provide personal information and ask for a medical visit. After the visit, if *va* exits without paying, a staff agent (*sa*), that is in charge of managing the visits on behalf of the hospital organisation, provides an enforcement to the visitor, namely he/she sends a fine at visitor's home. In *M*OISE functional description a "patient" mission $m1$ can be defined in terms of the following activities: *(i)* open the door, *(ii)* book for a visit, *(iii)* pay at the billing machine (optional), and *(iv)* exit. The staff agent holding the organisation, is indeed committed to the mission $m2$ by which he/she: *(i)* recognise patients who exit without paying, *(ii)* use the terminal to automatically send the increased fee at patient address.

To control social dynamics, the organisation adopts an organisational specification, which is fed to the OAs as *Norm-Org-Exp* automatically translated from *M*OISE description. Let us assume that, to support these patterns, the organisation has introduced

$N01$: $players(patient, vgroup, V) \wedge (V \geq 10)$	$\rightarrow fail(N01)$
$N02$: $committed\text{-}mission(patient, m1)$	$\rightarrow obligation(patient, achieve\text{-}goal(book\_visit))$
$N03$: $committed\text{-}mission(patient, m1)$	$\rightarrow obligation(patient, achieve\text{-}goal(pay))$
$N04$: $committed\text{-}mission(patient, m1)$	$\rightarrow obligation(patient, achieve\text{-}goal(exit))$
$N05$: $committed\text{-}mission(staff, m2)$	$\rightarrow obligation(staff, achieve\text{-}goal(add\text{-}fee))$
$N06$: $violated(N03)$	$\rightarrow obligation(staff, commit\text{-}mission(m2))$
$visitDoor$: $a\text{-}op\text{-}ev(enter(\$a), op\_exec\_start)$	$\rightarrow GroupBoard : adoptRole(\$a, patient)$
$visitDoor$: $a\text{-}op\text{-}ev(exit(\$a), op\_exec\_complete)$	$\rightarrow GroupBoard : leaveRole(\$a, patient)$
$GroupBoard$: $a\text{-}op\text{-}ev(adoptRole(\$a, patient), op\_exec\_complete)$	$\rightarrow SchemeBoard : commitMission(\$a, m1)$
$billingMachine$: $a\text{-}op\text{-}ev(signal, receipt)$	$\rightarrow SchemeBoard : setGoalAchieved(\$a, pay)$
$billingMachine$: $a\text{-}op\text{-}ev(signal, receiptAndFee)$	$\rightarrow SchemeBoard : setGoalAchieved(\$a, add\text{-}fee)$
$visitDoor$: $a\text{-}op\text{-}ev(exit(\$a), op\_exec\_complete)$	$\rightarrow SchemeBoard : setGoalAchieved(\$a, exit)$
$terminal$: $a\text{-}op\text{-}ev(signal, \text{"}sent\text{"})$	$\rightarrow SchemeBoard : setGoalAchieved(\$a, send\text{-}fee)$
$SchemeBoard$: $a\text{-}op\text{-}ev(commitMission(\$va, m1), op\_exec\_complete)$	$\rightarrow visitRoom : disable(\$va, exit)$
$SchemeBoard$: $a\text{-}op\text{-}ev(setGoalAchieved(\$sa, send\text{-}fee, \$va), op\_exec\_complete)$	$\rightarrow visitRoom : enable(\$va, exit)$
$NormBoard$: $a\text{-}prop\text{-}ev(property\_updated, N01, true)$	$\rightarrow visitRoom : disable(\_, enter)$
$NormBoard$: $a\text{-}prop\text{-}ev(property\_updated, N01, false)$	$\rightarrow visitRoom : enable(\_, enter)$

Fig. 3. *Norm-Org-Exp* (norm related) and *Org-Emb-Rules* (respectively *Count-as* and *Enact*) regulating the Hospital Scenario

particular environment artifacts EAs, which are deployed and controlled by the organisation in order to instrument the work environment: a "visit" door, a billing machine and a terminal, as they are detailed below:

```
artifactName: visitDoor, env.door
uic (operations): enter, exit
a-obs-prop: n_visitor
a-op-ev: enter,(op_exec_start)
         enter,(op_exec_complete)
         exit,(op_exec_start)
         exit,(op_exec_complete)
a-prop-ev: n_visitor, (prop_changed)
```

```
artifactName: terminal, env.terminal
uic (operations): send
a-op-ev: send, (op_exec_start)
         send, (signal, "sent")
         send, (op_exec_complete)

artifactName: billingMachine,env.billingMachine
uic (operations): pay, payFee
a-op-ev: pay, (op_exec_start)
         pay, (signal, receipt)
         pay, (op_exec_complete)
```

As showed in Fig. 3, the organisation (OAs) has been programmed according to a set of norms (i.e., *Norm-Org-Exp*). *N01* indicates a regimentation for the cardinality of the patient role by stating that no more than N agents can play the role *patient* in the group *vgroup*. Besides, obligations (O) have been introduced for regulating mission commitment (*N02-N05*) and for defining consequences of norm violation (*N06*). Namely, the norm *N03* states that a visitor agent committing to $m1$ is obliged to achieve the goal pay, while the norm *N06* states that, since a violation of *N03* occurs, a staff agent is obliged to commit to the mission $m2$.

Besides normative specification, Fig. 3 also indicates the *Org-Emb-Rule* that the organisation has specified to functionally link OAs to EAs artifacts. In this case, *Count-as* effects are defined to indicate how, since a noticeable event occurring in the overall system (*fact*), some additional outcomes need to be elicited at the organisational level. In so doing, an event *op_exec_start* dispatched by the door once a given agent is using it to enter, from the point of view of the organisation "count-as" adopting the role "patient". This role-adoption event suddenly "count-as" committing to mission $m1$, while

using the door to exit “count-as” leaving the role “patient”. At the same time, the event signalling that the staff agent has successfully used the terminal to send the fee to a given patient “count-as” having achieved the goal send-fee. It is worth remarking that events triggering count-as rules can rise either from EAs (as in the case of events occurring in the door and in the terminal) either from OAs (as in the case of role adoption, which is suddenly related to a mission commitment). Besides count-as, *Enact* effects are defined to indicate how, from the events occurring at the organisational level, some control scheme need to be applied to the environmental infrastructure. Notice that by not respecting the specified norms, the `NormBoard` OA automatically dispatches events indicating observable property change, which in short show ongoing violations. Violations are thus organisational events (*org-fact*) which suddenly elicit the application of some *Enact* rule which can be exploited for regimenting the environment. For instance, regimentations are installed by the organisation thanks to the enact rules stating that a door has to be disabled until a patient has paid the visit or the staff agent has sent the fee through the terminal. Besides, once the maximum cardinality for the role *patient* is reached an inconsistency is elicited, indicating the physical impossibility to further play that role. According to the norm $N01$, this result in an observable property update on the *NormBoard* (i.e., $fail(N01)$). Therefore, adopting the role patient when visitors cardinality is reached is an event enacting the removal of the `enter` operation from door *uic* (agents are no longer able to use it).

6 Conclusion

We described a computational model enabling agents that are not aware of organisational structures to operate in embodied, artifact based organisations. As a consequence, agents result situated in their application domain and can achieve their goals reasoning on native constructs as internal events and mental states. Besides, the model allows agents unaware of complex organisational specification to participate at complex organisational patterns. Moreover, thanks to the event based model of artifacts, organisations can be automatically acknowledged about relevant fact and ongoing changes in the work environment, thus easing the agents from the need to explicitly notify the systems about their ongoing activities. We envisage this aspect as a contribute for openness in MAS organisations and, as discussed in [3], a first step for the challenge of reconciling emergence with cognition, intentional behavior with social function, planning agents with unaware styles of cooperation/coordination.

References

1. http://cartago.sourceforge.net
2. Campos, J., Lòopez-Sànchez, M., Rodrìguez-Aguilar, J.A., Esteva, M.: Formalising Situatedness and Adaptation in Electronic Institutions. In: Proc., COIN 2008 (2008)
3. Catelfranchi, C.: Engineering Social Order. In: Omicini, A., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2000. LNCS (LNAI), vol. 1972, pp. 1–18. Springer, Heidelberg (2000)
4. Dastani, M.M., Grossi, D., Meyer, J.-J.C., Tinnemeier, N.A.M.: Normative Multi-Agent Programs and Their Logics. In: Proceedings, KRAMAS 2008 (2008)

5. Grossi, D., Aldewered, H., Dignum, F.: Ubi Lex, Ibi Poena: Designing norm enforcement in e-institutions. In: Noriega, P., Vázquez-Salceda, J., Boella, G., Boissier, O., Dignum, V., Fornara, N., Matson, E. (eds.) COIN 2006. LNCS (LNAI), vol. 4386, pp. 101–114. Springer, Heidelberg (2007)
6. Hewitt, C.: Perfect Disruption:The Paradigm Shift from Mental Agents to ORGs. IEEE Internet Computing 13 (2009)
7. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting Multi-Agent Organisations with Organisational Artifacts and Agents. Journal of Autonomous Agents and Multi-Agent Systems (2009)
8. Hübner, J.F., Sichman, J.S., Boissier, O.: Developing Organised Multi-Agent Systems Using the MOISE+ Model: Programming Issues at the System and Agent Levels. Agent-Oriented Software Engineering 1(3/4), 370–395 (2007)
9. Fabio, Y., Okuyama, R.H., da Rocha Costa, A.C.: A distributed normative infrastructure for situated multi-agent organisations. In: Proc., AAMAS 2008 (2008)
10. Ricci, A., Piunti, M., Acay, L.D., Bordini, R., Hubner, J., Dastani, M.: Integrating Artifact-Based Environments with Heterogeneous Agent-Programming Platforms. In: Proceedings of AAMAS 2008 (2008)
11. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in CArtAgO. In: Multi-Agent Programming: Languages,Platforms and Applications, vol. 2. Springer, Heidelberg (2009)
12. Ricci, A., Viroli, M.: A Formal Model for Artifact-Based Environments in MAS Programming. In: Programming Multi-Agent Systems, PROMAS 2009 (2009)
13. Searle, J.R.: The Construction of Social Reality. Free Press (1997)

MACSIMA: On the Effects of Adaptive Negotiation Behavior in Agent-Based Supply Networks

Christian Russ[1] and Alexander Walz[2]

[1] Dacos Software GmbH, Science Park 2, D-66123 Saarbrücken, Germany
christian.russ@dacos.com
[2] University of Stuttgart, Graduate School of Excellence Advanced Manufacturing Engineering, Breitscheidstr. 2c, D-70174 Stuttgart, Germany
alexander.walz@GSaME.eu

Abstract. In this paper, we describe the multiagent supply chain simulation framework MACSIMA that allows the design of large-scale supply network topologies consisting of a multitude of autonomous agents. MACSIMA provides all agents with an adaptive negotiation module providing the fine-tuning of learning capabilities on the basis of genetic algorithms as well as of settings controlling the exchange of information about finished negotiations with other cooperating agents. On this basis the co-evolution and adaptation of price negotiation strategies as well as coalition formation processes of self-interested business agents in B2B-networked domains can be fine-tuned, simulated and evaluated. Our evaluation of the effects of self-adaptation on the overall turnover and profit of a five-tier supply network scenario shows that coordination outcome and efficiency vary significantly in dependence on (a) the elaborateness of used learn parameterizations, (b) on the homogeneity of the distribution of learn parameter settings within the agent society and (c) on the information exchange settings. System outcomes are measured on a macro-level in overall turnover, profit, and communication efficiency as well as on a group-level in the distributed group portions on turnover and profit. Our analysis shows that an expert learn and information exchange parameterization of the agents results in an increase of the overall system outcome in sales and profit by approx. 500 percent and thus has to be fine-tuned for reaching an efficient and effective coordination outcome. Moreover, expert parameterizations result in an improved communication efficiency and outcome stability on the macro- and the group-level. Providing a subgroup of agents with superior learn capabilities results in a shift of sales and profit to the smarter agents.

Keywords: Distributed Artificial Intelligence, Multiagent Systems, Simulation Modeling and Output Analysis, Intelligent Agents, Evolutionary Learning, Experimental Economics, Agent-based Supply Chain Management, Coordination Mechanism Design, Bilateral Negotiation, Genetic Algorithms.

1 Introduction

A *supply chain* is a chain of possibly autonomous business entities that collectively procure, manufacture and distribute certain products. Since today's markets are highly

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 128–140, 2009.

dynamic, supply chain partner companies are forced to form supply chains on the basis of more flexible and co-operative partnerships. As a result, dynamic B2B supply networks evolve in which the partner companies are not fixed but partner selection and supply transactions are negotiated iteratively and dynamically over electronic B2B marketplaces providing facilities for posting offered or demanded materials, products, and price ideas as well as for matchmaking. But the coordination of the numerous resulting selection and negotiation processes often becomes too complex for humans to be handled efficiently. To cope this dilemma several researchers, e.g. Chaib-draa and Müller [1], suggest the use of intelligent agents acting on behalf of the supply chain companies and being able to adapt themselves to varying circumstances and partner companies quickly and efficiently. But so far there exist almost no consolidated findings about what coordination efficiency results if negotiations about the exchange of goods in a large-scale B2B scenario are totally transferred to business agents that are purely self-interested and thus try to outperform their partners by elaborated negotiation strategies. To examine the resulting dynamics within a supply network built up by adaptively negotiating agents, we have instantiated MACSIMA (**M**ulti **A**gent Supply **C**hain **SIM**ul**A**tion Framework).

2 The MACSIMA Framework

The MACSIMA framework has been implemented in Java and offers a set of generic supply chain agent types for instantiating supply network scenarios with the ability to take part in bilateral negotiations, as well as with elaborated learning capabilities, enabling them to learn simultaneously from negotiation successes and failures.

2.1 Simulation Scenario Definition and Agent Types

In MACSIMA, simulation scenarios can be defined and parameterized in a very detailed way as well as with a high degree of freedom, so that the user is totally free to decide on the structure of topology graphs, the number of tiers and the number of interacting agents on each tier. In our simulation runs conducted so far we have concentrated on different instantiations of a five-tier-supply-network for computer manufacturing, as sketched in figure 1.

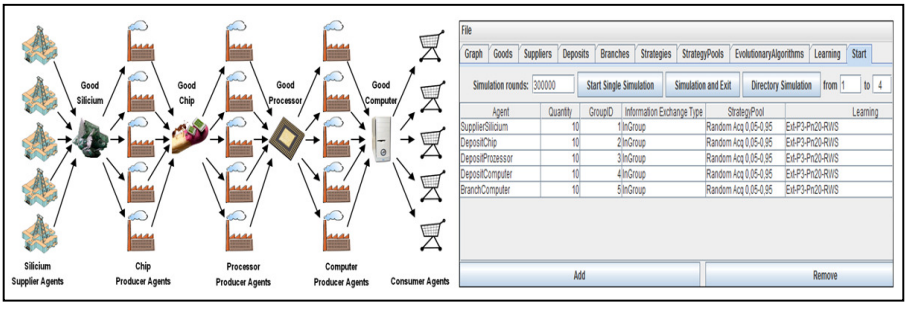


Fig. 1. Scenario for computer manufacturing with the corresponding top-level GUI

MACSIMA also offers a graphical user interface (GUI) that not only simplifies the definition of topologies but also enables one to parameterize the learning capabilities of each agent that is instantiated for a simulation run in detail as described later. The system designer can choose different agent types and their number to be instantiated. He can assign them to different groups and set their information exchange as well as their learning capabilities. The generic agent types offered by MACSIMA are:

1. *resource or supplier agents* (R_i) supply raw materials to the network.
2. *producer agents* (P_i) buy raw materials or semi-finished goods from other agents as input goods to a production function and offer their output goods for purchase.
3. *consumer agents* (C_i) buy products from producer agents and have a consumption function that specifies their maximal willingness to pay.
4. *good agents* (G_i) keep an account of the number of successful (transactions) or unsuccessful (rejections) price negotiations concerning a specific good.

Further views for specifying the supply network *graph*, the available *goods*, the negotiation *strategies*, the initial *strategy pools*, the available *evolutionary algorithms* and *learning* capabilities etc. can be accessed via clicking on the corresponding tabs.

The instantiated agents find each other by a directory service that administrates all the agents with their unique names, addresses, agent type and properties. Supply agents use this directory to find other agents with possibly intersecting goals. To find out if a mutually beneficial transaction can be carried out, each agent may select another agent in the network for starting a bilateral negotiation.

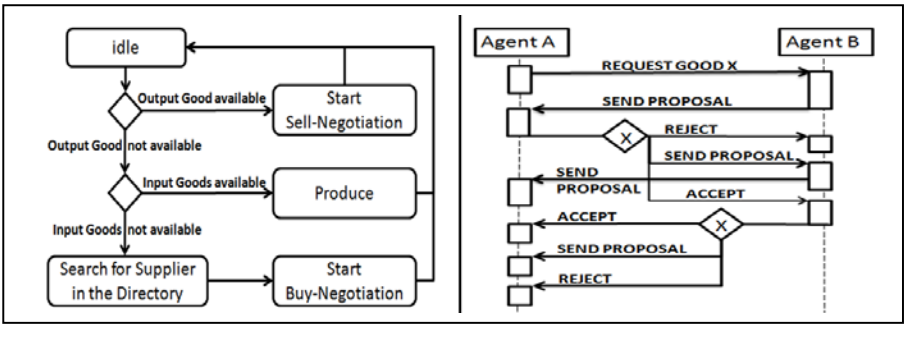


Fig. 2. Control cycle of a MACSIMA agent and its bilateral price negotiation protocol

2.2 Negotiation Protocol and Strategy Parameters

In the bilateral negotiation process, all agents are provided with the same action alternatives derived from Pruitt's strategic choice model [4] which states that humans select in every negotiation step from among five basic strategies (*unilateral concession*: *competitive behavior*, *coordinative behavior*, *idleness*, and *demolition)*. [2] states that this set can be further reduced to three negotiation action alternatives:

1. ***Accept*:** the price proposal of another agent is accepted and the transaction is conducted. The buyer pays the end price to the seller and receives the product.

2. ***Propose*:** the agent at turn does not agree to the price proposal of his opponent and makes a new proposal on his part (with or without a change to his last proposal).
3. ***Reject*:** an agent breaks off the negotiation and starts another negotiation.

On the basis of these three negotiation acts the agents in MACSIMA use the negotiation protocol sketched in figure 2 that is sufficient for modeling bilateral price negotiations between agents in the examined supply network application domain.

For modeling complex strategic negotiation behavior on this basis we use six *strategy parameters* that are explained in detail in [6] and determine the negotiation strategy of an agent. These can take on values from the interval [0;1] and are stored per agent in a so-called ***genotype***, a data structure suitable for processing by a genetic algorithm:

1. *acquisitiveness (A)*
2. *delta_change (DC)*
3. *delta_jump (DJ)*
4. *satisfaction (S)*
5. *weight_memory (WM)*
6. *reputation (R)*

The acquisitiveness specifies the willingness of an agent to make a unilateral concession on the next move. The price distance between the last price proposal and counteroffer of an opponent is specified by the parameter *delta_change*. The margin between the buying costs for input goods of an agent and the demanded selling price is defined by the *delta_jump* parameter. According to the next parameter *satisfaction* the agent checks in each round if the negotiation goes on or is aborted. Each agent calculates an internally "*sensed*" market price (SMP) to avoid nonsensical behavior and extortion offers. Therefore the agent stores the end prices of successful negotiations in a data structure *memory* and calculates the SMP with the parameter *weight_memory* and exponential smoothing according to:

```
memory = offeredPrice * weight_memory + memory * (1-weight_memory);
```

All counterproposals between the SMP and its double value are estimated as uncertain and a possible negotiation abort is tested with the formula:

```
if (offeredPrice >= memory) {
// ...then random reject check
if (randomNumberIsHigherThan(p_satisfaction)) { reject = true; }
  // reject all offers more than double memory
if (memory != 0 && offeredPrice > 2 * memory) { reject = true; } }
```

All proposals exceeding the doubled SMP are rejected directly to avoid extortion offers. The last parameter reputation specifies the probability of finishing a deal correctly. This leads to the strategy vector <*A, DC, DJ, S, WM, R*>.

2.3 Adaptive Negotiation Module and Learn Parameter Settings

Each negotiation module of an agent possesses a *genetic pool* of genotypes. This pool contains numerous genotypes that are employed in negotiations. After a negotiation has been finished a *fitness value* is calculated for the genotype depending on the negotiation outcome. Then the genotype is stored in combination with the ascertained

fitness value as *plumage* in a data structure called *population*. The sizes of pool and population can be flexibly set for the negotiation module of each agent.

After the start of a bilateral negotiation, the first step of an agent is to choose a genotype - determining his strategy for this negotiation - out of his pool of genotypes. Then, both agents negotiate until one of them aborts the negotiations or their price proposals cross and they make a mutually beneficial deal. After a successful negotiation both agents calculate a fitness value for the used genotype and store the resulting combination of genotype and estimated fitness as a so-called plumage into the population data structure. If their information exchange mode is set to external or mixed, they will afterwards send the resulting plumage to other agents, receive plumages from other allied agents and store the self-calculated as well as the received plumages in their population. If the number of stored plumages is larger than the population size the agents will start their learning process by using their individually parameterized evolutionary learning mechanism.

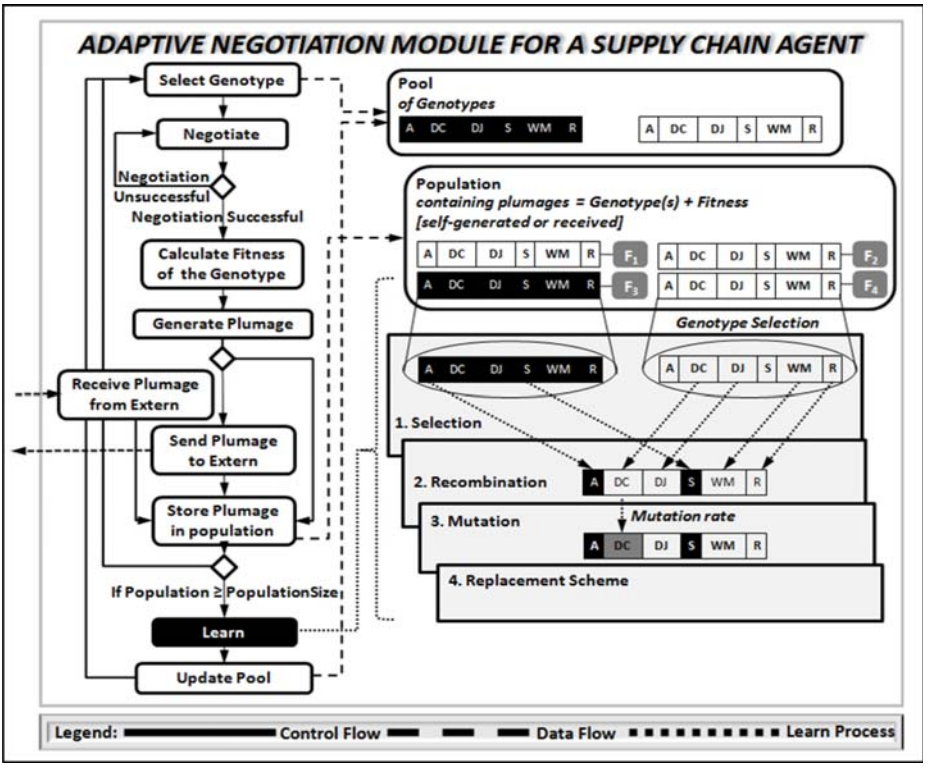


Fig. 3. Co-action of negotiation, information exchange, and learn process

When the learning process is started all plumages within the population are assigned to a *selection method*, which selects the plumages with the best fitness values and assigns as many plumages to a *recombination process* as the pool size allows. In this recombination process selected genotypes are recombined to new genotypes. Optionally, the newly built genotypes can be modified by probabilistic *mutation* after

the recombination step. In the last step of the learning process, the old population of the agent is deleted and the newly generated pool is assigned to the agent according to his specific replacement scheme. After that the agent may start new negotiations. The possible parameter settings for the learning process are described in table 1. The detailed parameterization is described in [6] and [7]. For this reason, we describe in the following only some selected parameters settings: The agents can learn either by themselves, i.e. do not exchange and use information with others or send and receive so called plumages (genotypes used in successfully finished negotiations with a fitness value) to other agents. In the external learning mode they learn exclusively from the experience made by "colleagues" in the mixed learning mode.

Table 1. Parameter settings for the learn process instantiation in MACSIMA

Information Exchange Settings	**Pool and Population Settings**
– Internal – External – Mixed	– Pool size – Population size
Fitness Calculation Methods	**Selection Methods**
– Price minus average (PMA) – Percental average proceeds (PAP) – Percental absolute proceeds(PAB) – Percental mean proceeds (PMP)	– Binary competition – Roulette-wheel-selection – Deterministic selection – Deterministic average selection – Deterministic average selection with deleting the worst individual
Recombination Methods	**Mutation Methods**
– N-point-crossover – Random crossover – No recombination	– Single mutation – General mutation
Replacement Scheme Settings	
– Elitism – Weak elitism	

Fitness Calculation

The fitness value can be calculated with several different methods. The simplest method is the *price minus average (PMA)* function where the fitness value is calculated as *fitness = average_price – current_price.* The *percental average proceeds (PAP)* method takes the duration of the negotiation in account by dividing the *PMA* value by the *average_price* (negotiated by using the genotype) times the number of rounds in which the genotype was used. The *percental absolute proceeds (PAB)* method replaces the *average_price* by a fixed *basic_price*. The *percental mean proceeds (PMP)* method uses the mean value of the starting price proposals of both negotiation partners (called *medium_price*) instead of the average price.

Selection

The *binary competition (BC)* compares two individuals I_k and I_l randomly selected from the current population and copies the one with the higher fitness value in the new population until its maximum size is reached. The *roulette wheel selection (RWS)* assigns a section on a wheel based on the fitness value according to the formula:

$$\alpha = 360 \frac{f(I_k)}{\sum_{n=1}^{N} f(I_n)}$$, with

α : Angel assigned to the k^{th} individual, N: Number of individuals

$f(I_k)$: Fitness value of the k^{th} individual

The *deterministic selection (DS)* method calculates an expectation value according to:

$$Expectation\ E(I_k) = f(I_k) \frac{N}{\sum_{n=1}^{N} f(I_n)}$$

The *deterministic average selection (DtS)* calculates for all genotypes in the current population an average fitness value *(AFV)* – in case a genotype was used several times. The genotype with the best *AFV* is copied directly into the new population, the genotype with the worst *AFV* is deleted and the remaining genotypes are selected according to the *DS* method. The *deterministic average selection (DAS)* fills up the new population by the genotypes from the current population with the *pool_size - 1* best average fitness values. The last free place in the new population is filled by a genotype whose genes are calculated as the mean of the genes of the genotypes that have already been copied into the new population. Afterwards, the *recombination process* breaks two genotypes at a time apart on *n* genes and links the resulting pieces cross-over. To keep the diversity the resulting set of recombined genotypes may be modified by several mutation methods before being copied into the new population.

3 Differentiation Factors of MACSIMA

An advantage of MACSIMA is the ability to set up nearly all possible supply network layouts and to instantiate them with numerous intelligent agents using a ***learning mechanism*** that can be adjusted to great detail. The experimental results of Lau et al. [3] show that such an evolutionary learning approach for adaptive negotiation agents in e-business scenarios can outperform a theoretically optimal negotiation mechanism guaranteeing Pareto optimality.

This is a progressive step as compared to the limited learning features of precedent approaches for the simulation and analysis of the effects of learning on the outcome of negotiations in such environments. They offer only very limited learning capabilities, e.g. Eymann [2] and Smith and Taylor [6], and almost no support in examining the effects on the generation of social welfare quantitatively. One further differentiating factor consists in the fact that MACSIMA agents can use different ***information exchange modes*** with respect to the extent of information exchange with other agents. In this way, the experimenter is able to build cooperating groups out of several agents on each tier and to examine the effects of coalition formation between agents. In this way, the evolution of an agent's negotiation strategy is not only guided by its own experience but can also take the experience of other agents into account.

Both allows for comparing different learning mechanisms in combination with different information exchange modes under the same external influences and constraints in a supply network. To our knowledge MACSIMA represents so far the only simulation framework suitable for this kind of experimental simulation and outcome analysis.

4 Simulation Results

The agents in MACSIMA log their internal status as well as their activities, negotiation steps and final outcomes in several separated and statistically evaluable data files. This raw data comprises information about the evolution of the agents' individual negotiation strategies, the course of negotiations together with their outcomes etc. and can be easily transformed into diagrams that show the course of the evolution process of the agents' strategy parameters together with the emerging price fluctuations for the traded goods in the time elapsed. In our simulation runs conducted so far we have mainly concentrated on different instantiations of a five-tier-supply-network for computer manufacturing as shown in figure 1. Scenarios have been run with 50 agents each on an Intel Quadcore-architecture with 3 GB RAM and 32bit Windows Vista as operating system. We have examined whether there exists a parameterization by which - if applied by all the agents in a network - social welfare maximizing effects may be expected.

4.1 Parameter Settings for Maximizing Profit, Turnover and Communication Efficiency

Therefore we have defined 50 simulation scenarios including two "baseline" scenarios each with a different parameterization of the learning process. In the first baseline setting learning was turned off for all agents and in the second the STDEA mechanism of Smith and Taylor [6] was used by all agents. An infelicitous parameter choice results in a waste of welfare in such a way that overall turnover and sales may

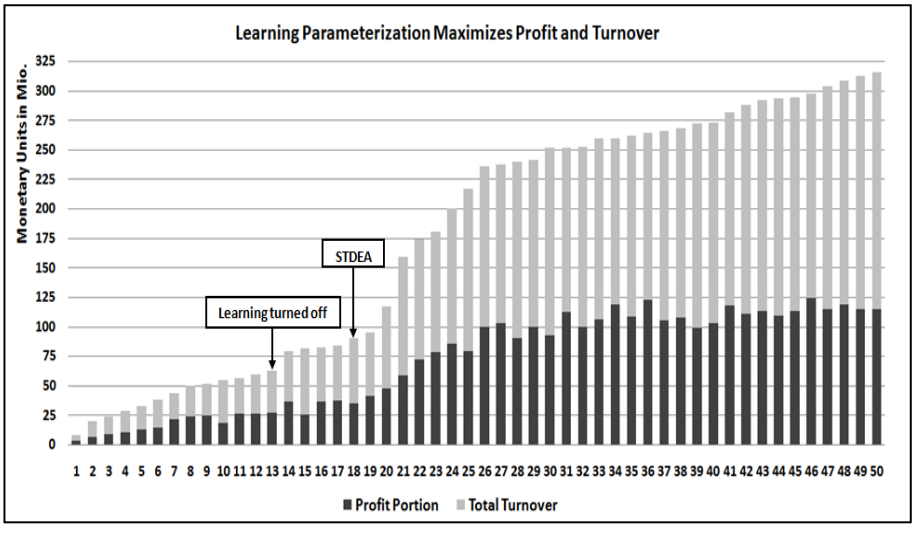


Fig. 4. Expert parameterization maximizes overall profit and turnover

fall under the level of both baseline settings (see fig. 4). Otherwise, an expert parameterization outperforms the first baseline by approximately 500 percent. The top parameterization we found (scenario 50 in the following figures) has the settings:

<pool_size = 3, population_size = 40, information_exchange = mixed, selection_method = roulette-wheel, recombination = n-point-crossover, mutation_rate = 0.5, Gaussian_width = 0.01, replacement_scheme = elitism>.

Due to the non-deterministic behavior of the agents during each simulation run, the values shown in fig. 4 represent the means of the overall sales and profit generated in 4 simulations runs that have been conducted for each of the 50 scenarios.

Profit and turnover correlate with the number of successfully finished negotiations, i.e. transactions (see fig. 5). This is due to the fact that an expert parameterization of the learning mechanism results in less negotiation break-offs since the agents adapt their negotiation behavior better to the expectations of their opponents.

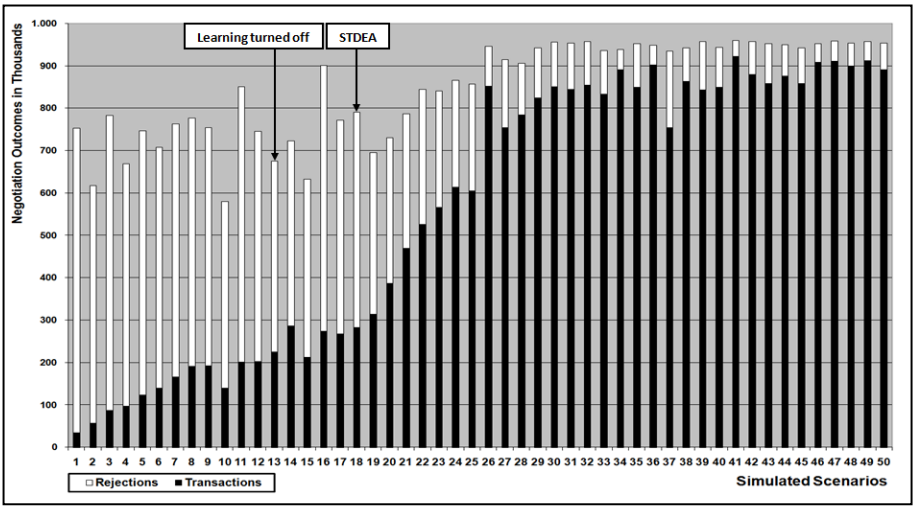


Fig. 5. Number of rejections and transactions in 50 scenarios

4.2 Expert Parameterization Improves Outcome Stability

The learning parameterizations generating the best outcomes with respect to sales and profit not only achieve this in the average but also show significantly fewer variations than the low-performing learning parameterizations of the scenarios 1 – 25.

This can be seen in fig. 6 showing the *variation coefficients (= root mean square deviation / mean)* for overall sales and profit calculated from 4 simulation runs for each scenario. But advanced learning parameterization not only produces more stable system outcomes on a macro-level, but also stabilizes the system outcomes on a medio-level, i.e. for the different interacting agent groups. Our analysis shows e.g. that the variations of the sales and profit portions of the agent groups decrease with the performance of the learning mechanism as can be seen in fig. 7 showing the variation of the group outcomes for 5 clusters with 10 scenarios each.

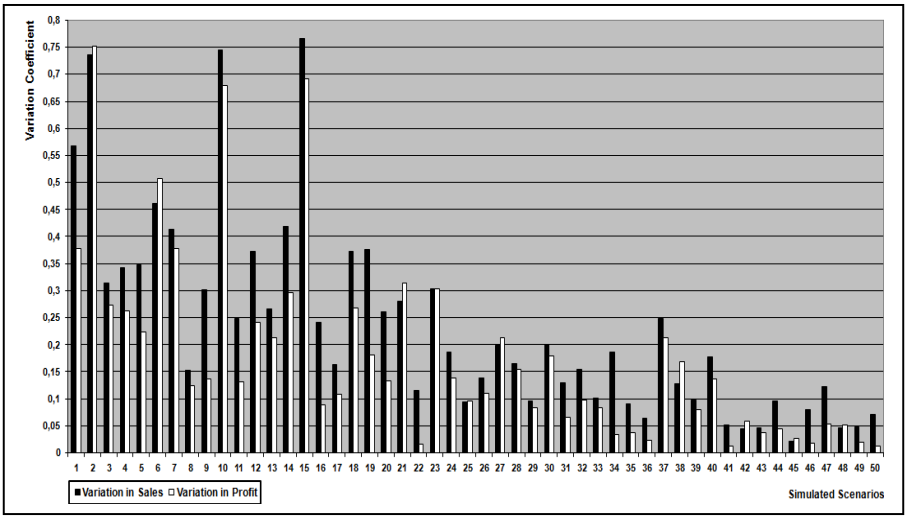


Fig. 6. Variation coefficients for overall sales and profit in 50 scenarios

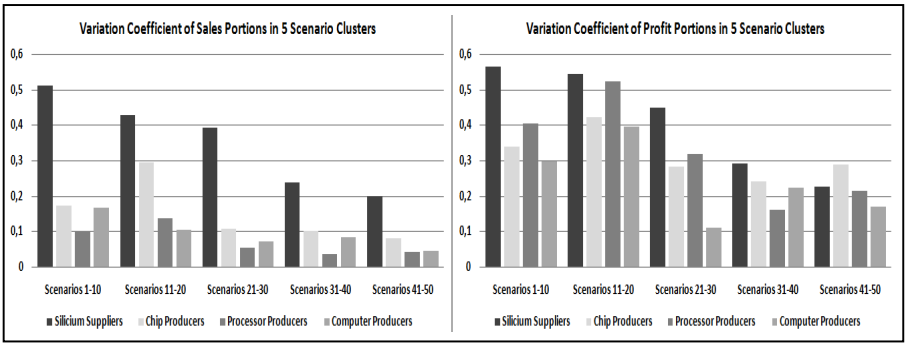


Fig. 7. Group sales and profit in 5 scenario clusters

This figure shows the results in a clustered manner for the sake of clarity. To underline the findings, we show in figure 8 also the detailed mean values (from 4 simulation runs) for the group sales and profits for the 10 lowest-performing (cluster 1) and the 10 best-performing (cluster 5) scenarios. By comparing the charts (A) and (B) as well as (C) with (D) it can be seen that the overall system outcome in sales and profit is distributed much more evenly in cluster 5 than in cluster 1.

4.3 Effects of a Heterogeneous Parameterization of the Agents

So far we have only analyzed scenarios in which all agents use the same learning parameterization. Now we examine the sales and profit distribution effects that occur when an agent group is composed of agents with different learning capabilities.

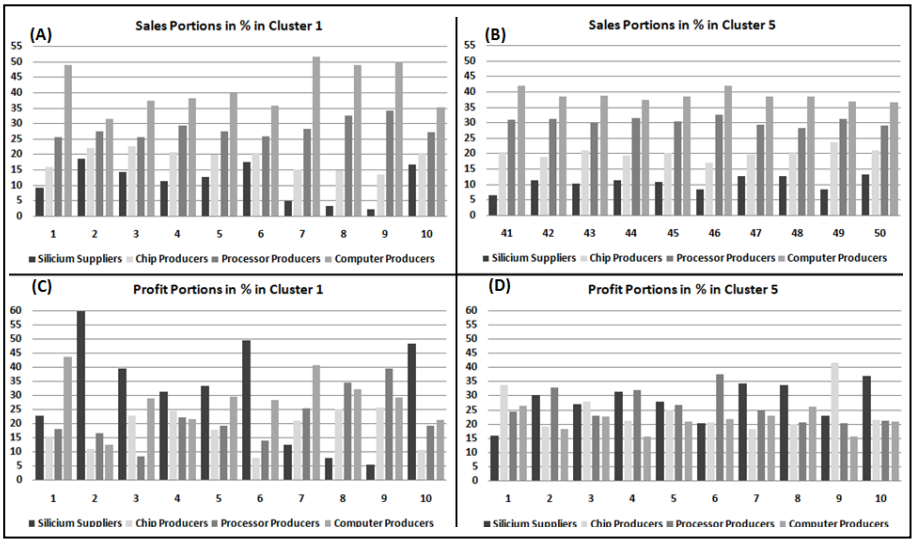


Fig. 8. Sales and profit distribution in the 10 poorest and richest scenarios

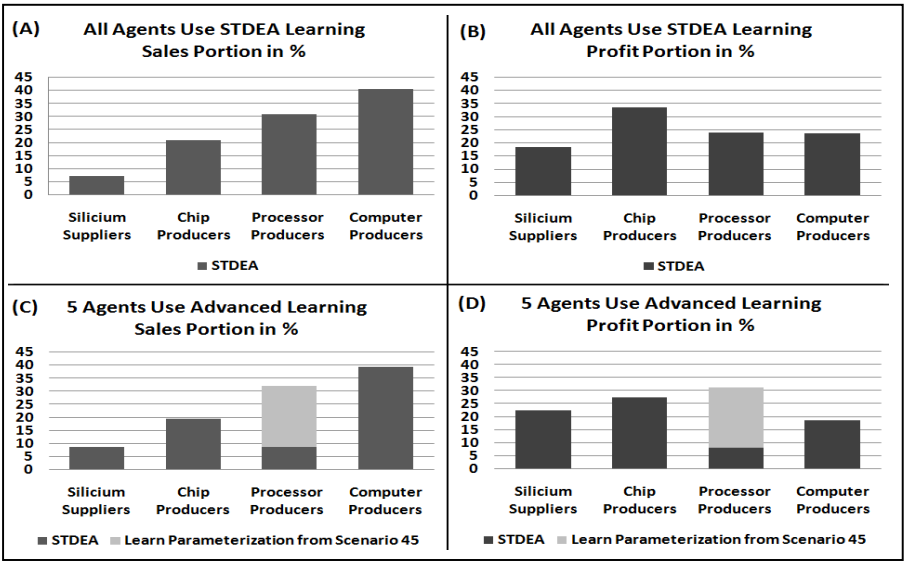


Fig. 9. Agents with expert parameterization outperform their neighbors

As shown in figure 9, we have conducted a test run in which we designed the learn capabilities of the 10 processor producer agents at level 3 of our 5 tier test supply chain heterogeneously. We provided 5 agents with the fifth-best learn parameterization used in scenario 45 from figure 4.

All other agents were provided with the STDEA parameterization from scenario 18. In figure 9 (C) it can be seen that this results in a slight increase in the collective sales portion of all processor producer agents at the expense of the neighboring chip and computer producers. Since the chip producers learn to make more concessions in negotiations with the processor producers they transfer this adaptation to their negotiation behavior in interactions with the upstream suppliers. The supplier agents take advantage from this effect and also manage to increase their sales portion slightly. In the overall sales portion of the processor producers, the 5 agents instantiated with the expert learn parameterization realize a significantly disproportionate share of 72%. Furthermore, figure 9 (D) shows that the 5 better parameterized agents realize a similar share of 73% in the overall profit portion of all processor agents. Compared to the setting in which all agents use the STDEA parameterization (see fig. 10 (B)) the group of processor agents increases their profit portion by 7 percentage points at the expense of significant losses at the neighbored tiers (chip producers -6 percentage points and computer producers -5 percentage points) while the supplier agents benefit from the "weakness" of the chip producers and increase their profit by 4 percentage points.

5 Conclusion and Outlook

We have described the MACSIMA framework for simulating the negotiation-based coordination of self-interested agents in B2B-enabled supply network domains. In particular, we presented a negotiation protocol for bilateral price negotiations together with the design of an adaptive negotiation module that can be used by the agents for adapting their negotiation strategies. We have outlined simulation results with a first focus on the effects of different learn mechanism parameterizations on the overall profit and sales of a networked agent-based supply economy. The results show that, if one intends to use negotiating agents for coordinating a supply network, the parameterization of the learning mechanism of the agents has to be fine-tuned for reaching an efficient and effective coordination outcome. Depending on the parameter settings of the agents the overall profit and turnover of a supply network varies significantly such that – compared to a scenario in which the agent have no learning capabilities (scenario 18) – an endowment of all agents with an expert learn parameterization can increase the overall system outcome by approx. 500 percent.

Moreover, an expert parameterization results in an improved communication efficiency as well as in an increased outcome stability with respect to absolute and distributed sales and profit results on the macro- and the group-level. Furthermore, endowing a subgroup of agents on one tier of the supply chain with superior learn capabilities results in a shift of sales and profit from neighbored tiers to the smarter agents. Further work will be directed towards a detailed analysis of the micro- and macro-effects (e.g. society loses welfare whereas some individuals benefit) in dependence on varying learn and information exchange parameterizations on different tiers of a supply chain. Beyond that, we intend to further improve the adaptation performance of the negotiation module by integrating a second pool so that adaptation processes for buy- and sell-negotiations can be conducted completely self-contained.

References

1. Chaib-draa, B., Müller, J.P. (eds.): Multiagent based Supply Chain Management. Springer, Berlin (2006)
2. Eymann, T.: AVALANCHE - Ein agenten-basierter dezentraler Koordinationsmechanismus für elektronische Märkte, Inaugural-Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau (2000)
3. Lau, R.Y.K., et al.: An Evolutionary Learning Approach for Adaptive Negotiation Agents. International Journal of Intelligent Systems 21(1), 41–72 (2006)
4. Pruitt, D.G.: Negotiation Behavior. Academic Press, New York (1981)
5. Russ, C., Walz, A.: MACSIMA: An Agent Framework for Simulating the Evolution of Negotiation Strategies in B2B-Networked Economies. In: Proceedings of the 23rd European Conference on Modelling and Simulation, ECMS 2009 (upcoming, 2009)
6. Smith, R.E., Taylor, N.: A Framework for Evolutionary Computation in Agent-Based Systems. In: Looney, C., Castaing, J. (eds.) Proceedings of the 1998 International Conference on Intelligent Systems, pp. S221–S224 (1998)
7. Weicker, K.: Evolutionäre Algorithmen. In: Leitfäden der Informatik, Frankfurt, B.G. Teubner, Stuttgart – Leipzig – Wiesbaden (2002)

Towards Reactive Scheduling for Large-Scale Virtual Power Plants

Martin Tröschel and Hans-Jürgen Appelrath

OFFIS - Institute for Information Technology, Escherweg 2, 26121 Oldenburg
martin.troeschel@offis.de
http://www.offis.de/

Abstract. Concerning distributed energy management, virtual power plants are a frequently discussed topic. Although there are several different approaches to the coordination of distributed energy resources in this context, the inherent dynamics of this complex task especially relating to reactive scheduling have mostly been neglected. As a consequence, this paper discusses MARS, a multiagent-based coordination approach contributing to the solution of the reactive scheduling problem for virtual power plants. Following an introduction to scheduling in the energy domain, a general formalization of scheduling in virtual power plants is given. This formalization is used as starting point for the specification of the control system coordinating the distributed energy resources. Finally, the performance of the resulting domain-specific multiagent system is reviewed by means of simulation.

1 Introduction

Virtual power plants (VPP), defined as capacity bundling of distributed energy resources (DER) by means of an IT-infrastructure, are expected to play an important role in the future energy supply. By application of different coordination mechanisms, they allow for combined market participation of energy resources such as combined heat and power (CHP) or photovoltaic (PV) plants, enable enhanced load levelling in distribution grids [7] and contribute to a reduction of the otherwise uncoordinated and stochastic electrical feed-in of distributed energy resources [8].

Centralized architectures with a single control unit currently are the predominant ([1], [7], [8]) approach, as they allow for a simple implementation of different optimization strategies. In contrast, recent works propose hierarchical and decentralized architectures based on multiagent systems focusing on single coordination strategies (q.v. Sec. 2). Generally, coordination strategies and mechanisms for the control of distributed energy resources can be classified on a temporal level. According to [3], two distinct scheduling modes can be distinguished: predictive scheduling, comprising resource scheduling tasks in long-term and short-term planning, and reactive scheduling. The goal of predictive scheduling, and day-ahead resource scheduling in particular, is the generation of optimal operation schedules for power plants. Optimality depends on the application of

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 141–152, 2009.

the power plant, e.g. optimal matching of electrical power supply and demand, profit maximization in energy trade, etc. The necessity for reactive mechanisms in (virtual) power plant control particularly arises from inevitable forecasting errors[1]. For conventional power plants like coal-fired power plants, day-ahead scheduling is primarily dependent on forecasts of electric power demand. In virtual power plants comprising a large number of distributed power units, however, the quality of day-ahead scheduling also depends on forecasts of electrical power supply, e.g. of PV plants, and of the estimated thermal demand of buildings with CHP plants. Thus, the overall inaccuracy of the data used as the basis for day-ahead operation scheduling increases, resulting in run-time deviations from the initial schedules of single DER and the overall schedule of the VPP. As a consequence of these schedule deviations, expensive balance energy has to be provided by conventional power plants, thus impairing both the economic and the ecological potential of distributed energy generation.

Dealing with schedule deviations is the main task of reactive scheduling. Upon detection of a schedule deviation, the control mechanisms of a virtual power plant has to provide alternative power capacities in order to mimimize the resulting deviation of the overall schedule[2]. The typically large number of controlled DER complicates this task: First, suitable rescheduling potential has to be identified. The individual potential of each DER depends on on-site and time-dependent state information like the charging level of the thermal water storage of a CHP plant. Second, single DER might not be able to provide sufficient rescheduling potential. In this case, multiple distributed plants have to be coordinated adequately.

Despite its importance, reactive scheduling for virtual power plants currently is a little considered research topic. Therefore, this work introduces a flexible reactive control system for VPP that contributes to a minimization of overall schedule deviations and supports the realization of a more dependable distributed electrical power supply.

2 Related Work

In [1], a central supervision and control component for online optimization in virtual power plants is proposed. The basic idea of online optimization is a continuous re-optimization of resource schedules, adjusting them to up-to-date information such as the resources' states, weather conditions, etc. Since the conceptual details are not explained, a final analysis regarding the suitability of this approach is not possible.

In contrast, the PowerMatcher approach discussed in [6] introduces a hierarchically organized multiagent system for the optimized matching of electrical supply and electrical demand. Consumers and producers of electrical energy are represented by individual agents, which are assigned to hierarchically superior SD-Matcher agents (q.v. Fig. 1). Supply-demand-matching is realized by means of an

[1] Both thermal and electrical demand depend on human behaviour.
[2] Q.v. Sec. 3 for a formalization of this requirement.

on-the-fly auction-based market mechanism, thus allowing for pareto-optimal results and avoiding the necessity of reactive mechanisms. The high quality of the resulting coordination of DER is a major advantage of the PowerMatcher approach. On the other hand, the focusing on a single coordination strategy restricts the potential applications of virtual power plants. For example, a directed provision of balance energy is not possible.

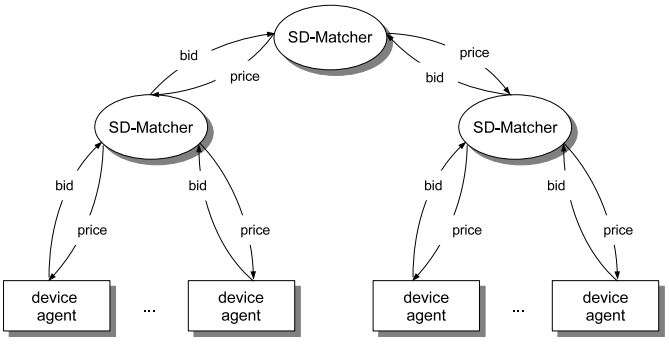


Fig. 1. Hierarchically organized supply-demand-matching according to [6]

Like PowerMatcher, DEZENT [9] is realized as a hierarchically organized multiagent system implementing an auction-based market mechanism. Since the control system of DEZENT aims at a real-time coordination of supply and demand of electrical energy, the main difference regarding the PowerMatcher approach is the temporal resolution of the auction cycles. Thus, the already identified advantages and disadvantages of PowerMatcher also apply to DEZENT.

3 Formalization of Scheduling for Virtual Power Plants

The following section presents a formalization of scheduling for virtual power plants. The terminology is a prerequisite for specification of the multiagent-based control system proposed in Sec. 4.

To begin with, a set of resources is introduced:

Definition 1 (Resources). *$R = \{r_1, ..., r_n\}$ is a final, non-empty set of distributed energy resources.*

Following the definition given in Sec. 1, a VPP supervises and controls all DER $r \in R$ by generating and adjusting operation schedules for a given scheduling horizon:

Definition 2 (Scheduling horizon). *The **scheduling horizon** $T \subset \mathbb{N}$ is a final, non-empty set of period identifiers. $t \in T$ is called **scheduling period**.*

Schedules assign a certain amount of scheduled electrical power to the given scheduling periods $t \in T$:

Definition 3 (Schedule). *For scheduling period $t \in T$ and scheduled power $p \in \mathbb{Q}$ the function $s : T \to \mathbb{Q}$, $s(t) = p$, is called* ***schedule****. S is the set of all schedules.*

Operation schedules assign schedules $s \in S$ to resources $r \in R$:

Definition 4 (Operation schedule). *For resource $r \in R$ and schedule $s_r \in S$ the function $os : R \to S$, $os(r) = s_r$, is called* ***operation schedule****. OS is the set of all operation schedules.*

The overall schedule of a virtual power plant comprises all individual resource schedules and can be derived from the associated operation schedule:

Definition 5 (Overall schedule). *For a scheduling horizon T, a set of resources R and an operation schedule $os \in OS$ the* ***overall schedule*** *of a VPP is defined by*

$$\hat{s}_{os}(t) := \sum_{r \in R} s_r(t) \text{ where } s_r = os(r) \text{ and } t \in T. \tag{1}$$

Predictive scheduling for virtual power plants particularly includes the task to generate an initial operation schedule. In doing so, several constraints regarding the resources' characterstics have to be taken into account:

Definition 6 (Hard constraints). *$HC = \{hc_1, ..., hc_h\}$ is a final set of* ***hard constraints****. The specification of HC depends on the application scenario of the VPP.*

In the context of distributed energy resources, hard constraints can be classified as time-invariant constraints and time-variant or rather state-dependent constraints. Examples for time-invariant constraints are technologically restricted maxmimum (and minimum) power supply capacities or minimum run-times of CHP plants. Examples for state-dependent constraints are the charging level of storage devices for thermal or electrical energy, the electrical power feed-in of PV plants or the thermal demand of buildings with CHP plants.

Definition 7 (Soft constraints). *$SC = \{sc_1, ..., sc_s\}$ is a final set of* ***soft constraints****. The specification of SC depends on the application scenario of the VPP.*

While hard constraints especially refer to technological attributes, soft constraints comprise economic and ecological preferences such as fuel costs or preferred usage of power supplies based on carbon dioxid-neutral energy sources. By now, day-ahead scheduling can be defined as follows:

Definition 8 (Day-ahead scheduling problem). *A* ***day-ahead scheduling problem*** *for virtual power plants is given by the 5-tuple (R, T, f, HC, SC). It comprises the task to find an operation schedule $os \in OS$ for a set of resources R and a scheduling horizon T w.l.o.g. maximizing a target function $f : OS \to \mathbb{Q}$ and complying with the constraints in HC and SC.*

As already mentioned in Sec. 1, deviations from the initially generated operation schedule are likely to occur. Therefore, reactive scheduling is necessary. The following definitions formalize this task.

Definition 9 (Partially realized schedule). *For a scheduling period $t \in T$ and a schedule $s \in S$, the **partially realized schedule** $\overline{s}$ at time t is definded by*

$$\overline{s}(t_i) := \begin{cases} s(t_i) + \alpha(t_i) & \text{if } t_i \leq t, \\ s(t_i) & \text{else.} \end{cases} \tag{2}$$

Partially realized schedules may exihibit positive and/or negative schedule deviations $\alpha(t_i)$ for each scheduling period $t_i \leq t, t_i, t \in T$, where t is the present period in terms of execution time.

Definition 10 (Schedule deviation). *For a scheduling period $t \in T$, a schedule $s \in S$ and the partially realized schedule $\overline{s}$ at time t the **schedule deviation** is defined by*

$$\delta_{(s,\overline{s})}(t) := \overline{s}(t) - s(t). \tag{3}$$

Schedule deviations may add up over time. The main goal of reactive scheduling in VPP is the minimization of these deviations. The metric introduced below is used to quantify the performance of reactive scheduling methods relating to individual resource schedules:

Definition 11 (Mean schedule deviation). *For a scheduling period $t \in T$, a schedule $s \in S$ and the partially realized schedule $\overline{s}$ at time t the **mean schedule deviation** is defined by*

$$\Delta_{(s,\overline{s})} := \sqrt{\frac{1}{|T|} \cdot \sum_{t \in T} (\delta_{(s,\overline{s})}(t))^2} = \sqrt{\frac{1}{|T|} \cdot \sum_{t \in T} (\overline{s}(t) - s(t))^2}. \tag{4}$$

The following definitions expand the concepts of partially realization and deviations from schedules to operation schedules.

Definition 12 (Partially realized operation schedule). *For a scheduling period $t \in T$ and an operation schedule $os \in OS$ the **partially realized operation schedule** $\overline{os}$ at time t is defined by*

$$\overline{os}(r) := \overline{s}_r \text{ for all } r \in R. \tag{5}$$

Partially realized operation schedules assign partially realized schedules to resources. Therefore, deviations from operation schedules are defined relating to the overall schedule of a VPP:

Definition 13 (Operation schedule deviation). *For a scheduling period $t \in T$, an operation schedule $os \in OS$ and the partially realized operation schedule $\overline{os}$ at time t the **operation schedule deviation** at time t is defined by*

$$\begin{aligned} \delta_{(os,\overline{os})}(t) &:= \delta_{(\hat{s}_{os},\overline{\hat{s}}_{os})}(t) \\ &= \sum_{r \in R} (\overline{s}_r(t) - s_r(t)). \end{aligned} \tag{6}$$

With this definition in mind, the following metric can be introduced to measure the performance of reactive scheduling methods relating to the overall schedule of a VPP:

Definition 14 (Mean operation schedule deviation). *For a scheduling period $t \in T$, an operation schedule $os \in OS$ and the partially realized operation schedule $\overline{os}$ at time t the **mean operation schedule deviation** at time t is defined by*

$$\Delta_{(os,\overline{os})} := \sqrt{\frac{1}{|T|} \cdot \sum_{t \in T} (\delta_{(os,\overline{os})}(t))^2} = \sqrt{\frac{1}{|T|} \cdot \sum_{t \in T} (\sum_{r \in R} (\overline{s}_r(t) - s_r(t)))^2}. \quad (7)$$

With the background of definitions Def. 1 - Def. 14, it is possible to define reactive scheduling for virtual power plants:

Definition 15 (Reactive scheduling problem). *The **reactive scheduling problem** for virtual power plants at execution time t is characterized by the 5-tuple $RSP = (R, t, \overline{os}, HC, SC)$. It comprises the task to find an operation schedule $\tilde{os} \in OS$ minimizing the mean operation schedule variance $\Delta_{(os,\overline{os})}$ of an initial operation schedule $os \in OS$ at time $t \in T$ by generating individual schedules for all resources $r \in R$ in compliance with the constraints in HC and (preferably) SC.*

It must be pointed out that Def. 15 is limited, as it neglects the target function of the day-ahead scheduling problem and therefore doesn't take an online optimization of the operation schedule into account. Therefore, an optimal solution $\tilde{os} \in OS$ of a reactive scheduling problem RSP will result in $\Delta_{(\tilde{os},\overline{os})} = 0$, thus avoiding unnecessary provision of balancing energy.

4 Reactive Scheduling for Virtual Power Plants

Based on Def. 1 - Def. 15, this section proposes MARS, a multiagent-based control system able to solve the reactive scheduling problem for virtual power plants. First, the domain-specific requirements concerning the coordination of DER in VPP are identified. Second, two types of agents, namely resource agents and order agents, are introduced. Last, the interactions necessary to resolve schedule deviations are specified in compliance with the identified requirements.

4.1 Domain-Specific Requirements

Multiagent-based control systems have been successfully applied to scheduling in different domains such as production planning and control [5] or transportation scheduling [4]. Yet, scheduling for VPP is subject to a set of distinct, domain-specific requirements:

– DSR_1: The spatially distributed character of virtual power plants results in location-dependent basic conditions for individual energy resources, e.g. local solar irradiation, that have to be taken into account.

- DSR_2: Depending on individual power capacities, for certain applications of VPP such as provision of balancing power the coordination of several thousand DER may be required.
- DSR_3: Future virtual power plants are not necessarily statically composed. Thinking of e-mobility and of electric vehicles as mobile storage devices for electrical energy in particular, the plant configuration of virtual power plants may change even during run-time.

Due to their inherent robustness and flexibility, distributed control systems based on intelligent agents are especially suitable to cover these requirements.

4.2 MARS – Multiagent-Based Reactive Scheduling System for Virtual Power Plants

Given an exemplary set of constraints, Table 1 lists the tasks necessary to solve RSP as well as the affected DER of a VPP:

Table 1. Tasks necessary for reactive scheduling in VPP.

ID	task	components
time-invariant hard constraints (e.g. regarding CHP plants or storage devices)		
T_1	adjust $\overline{s}_r$ such that $\forall t \in T : p_{r,min} \leq \overline{s}_r(t) \leq p_{r,max}$, where $p_{r,min}$ and $p_{r,max}$ are the minimum and maximum power capacities of ressource r	$r \in R$
T_2	adjust $\overline{s}_r$ in compliance with the minimum and maximum run-times of ressource r	$r \in R$
T_3	adjust $\overline{s}_r$ such that $\forall t \in T : c_{r,min} \leq \overline{s}_r(t) \leq c_{r,max}$, where $c_{r,min}$ and $c_{r,max}$ are the minimum and maximum charging levels of ressource r	$r \in R$
state-dependent hard constraints (e.g. regarding PV plants or storage devices)		
T_4	adjust $\overline{s}_r$ such that $\forall t \in T : p_{r,min} \leq \overline{s}_r(t) \leq p_r(t)$, where $p_r(t)$ is the current power capacity of ressource r	$r \in R$
T_5	adjust $\overline{s}_r$ such that $\forall t \in T : c_{r,min} \leq \overline{s}_r(t) \leq c_r(t)$, where $c_r(t)$ is the current charging level of ressource r	$r \in R$
soft constraints (e.g. economical, local optimization)		
T_6	balance the utilization of ressource r over scheduling horizon	$r \in R$
T_7	maximize the degree of utilization of ressource r	$r \in R$
global optimization		
T_8	minimize $\delta_{(s_r,\overline{s}_r)}(t)$ via negotiation of power capacities with alternative resources $r'_1, ..., r'_k$	$r, r'_1, ..., r'_k \in R$

Solving the reactive scheduling problem $RSP = (R, t, \overline{os}, HC, SC)$ at time t therefore comprises at least the following steps:

1. For each resource $r \in R$, check the compliance of the partially realized schedule $\overline{s}_r$ with all time-invariant hard constraints in HC (T_1 - T_3).

2. For each resource $r \in R$, check the compliance of the partially realized schedule $\overline{s}_r$ with all state-dependent hard constraints in HC (T_4 - T_5).
3. For each resource $r \in R$, try to minimize the schedule deviation $\delta_{(s_r,\overline{s}_r)}(t)$ by reassigning the required power capacity to alternative resources (T_8).
4. For each resource $r \in R$, try to comply with all soft constraints in SC (T_6 - T_7).

Since tasks T_1 - T_9 solely refer to a single resource $r \in R$ and can thus be solved locally, it is suitable to introduce a **resource agent** for every resource $r \in R$ of a VPP. A resource agent supervises it's dedicated DER and takes care of any schedule deviations occuring during execution time. Fig. 2 displays the internal states of a resource agent: First, the partially realized schedule $\overline{s}_r \in S$ of resource r is adjusted to comply with both time-invariant and state-dependent hard constraints in HC. Second, any schedule deviations resulting from the adjustments made in previous steps have to be minimized by negotiating power capacities with alternative resource agents. Third, $\overline{s}_r$ can be locally optimized regarding the soft constraints in SC.

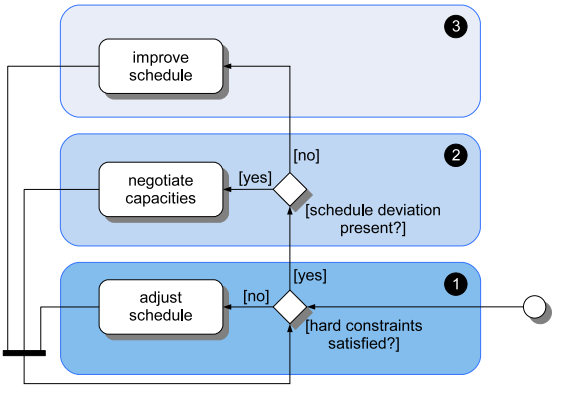


Fig. 2. Internal state diagram of a resource agent

As displayed in Fig. 2, the state transitions are organized in a prioritized manner. Thus, a resource agent will only try to optimize it's resource's schedule regarding SC if all hard constraints are fulfilled and all schedule deviations are handled. Furthermore, the resource agent ensures that schedule $\overline{s}_r$ is feasible and thus physically executable at any time. The successful handling of schedule deviations (state 2) can be subdivided into two tasks: First, promising resources have to be localized. Second, the required power capacity has to be negotiated. Since a VPP may comprise a large number of distributed energy resources, finding suitable negotiating partners is an independent problem (q.v. Sec. 6). Given the possibly large number of DER and the fact that day-ahead scheduling relies on several types of forecasts (q.v. Sec. 1), schedule deviations are likely to

occur. Furthermore, the handling of schedule deviations can be extensive and time-consuming. Thus, **order agents** solely responsible to complete task T_{10} are introduced. This yields the following specification of agent types:

Table 2. Specification of agent types of a distributed control system for VPP

agent type	available information	allocated tasks
resource agent	properties of resource $r \in R$, initial resource schedule $s_r \in S$, current resource schedule $\overline{s}_r \in S$,	T_1 - T_9
order agent	target power capacity $p_t \in \mathbb{Q}$, deadline $t_d \in T$, initiating resource agent	T_{10}

Based on the internal state diagram of resource agents in Fig. 2 and the specification of agent types in Table 2, the process displayed in Fig. 3 allows for a flexible and efficient[3] handling of local schedule deviations $\delta_{(s_{r_i}, \overline{s}_{r_i})}$.

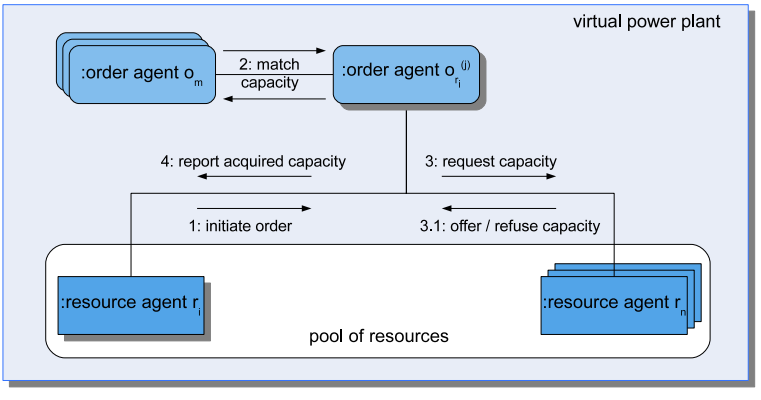


Fig. 3. Handling of local schedule deviations

As simplification, it is assumed that all resource agents and order agents mutually know each other. Thus, the search for negotiation partners can be momentarily omitted (q.v. Sec. 6). The handling of schedule deviations subdivides into five steps:

1. At execution time t, the resource agent supervising resource r_i detects a local schedule deviation $\delta_{(s_{r_i}, \overline{s}_{r_i})}(t)$. To handle this event, an order agent $o_{r_i}^{(j)}$ is initiated with the goal to acquire the power capacity $p_t = \delta_{(s_{r_i}, \overline{s}_{r_i})}(t)$ to balance the deviation.

[3] Q.v. Sec. 5.

2. Based on the Contract Net interaction protocol, the order agent $o_{r_i}^{(j)}$ negiotiates with other order agents in order to find a complementary target capacity. At best, $o_{r_i}^{(j)}$ (and one or more other order agents) acquires it's target capacity without the necessity of further negotiations with resource agents.
3. If $o_{r_i}^{(j)}$ hasn't reached it's target capacity yet, it enters Contract Net-based capacity negotiations with a set of resource agents selected from the known resources.
4. If the target capacity p_t is reached after the negotiations or the deadline t_d has expired, the order agent reports back to the initiating resource agent r_i, which adjusts it's schedule according to $o_{r_i}^{(j)}$'s successfully acquired capacity. Then, $o_{r_i}^{(j)}$ terminates.

An advantage of the resource and order agents' distributed competences is especially the possibilty to further specialize both agent types according to their distinct tasks. Plus, resource agents provide a continuous supervision of their associated distributed energy resource, since they don't have to proactively[4] negotiate with other agents. Concerning the domain-specific requirements identified at the beginning of this section, it has to be pointed out that especially the requirements DSR_1 and DSR_3 are covered due to the distributed character of the proposed control system. Requirement DSR_2 is subject to the discussion in Sec. 6.

5 Evaluation

5.1 Premises

The evaluation of MARS was realized using the simulation tool "E3Sim - Simulation of Distributed Energy Resources, Energy Infrastructures and Energy Management Systems", which was developed in an interdisciplinary effort by the "Energy Research Alliance Lower Saxony" [7]. E3Sim models CHP and PV plants, thermo-hydraulic properties of (non-business) buildings and power grid characteristics. Three different configurations of VPP with an increasing number N_{CHP} of CHP plants providing the total electrical power P_{el} were simulated (q.v. Table 3). The dimensioning of the simulated CHP units was based on the total yearly thermal demand of the respective building regarding an average yearly operation time of approximately 5000 to 6000 hours. All units were assumed to support modulating operation and were additionally equipped with a thermal storage able to buffer the thermal output of approximately two full load hours. As day-ahead scheduling method, the system-load oriented operation mode of CHP [7] was applied. As alternative reactive scheduling concept, a self-developed central dispatcher (CDisp) controlling all DER was implemented. CDisp handles schedule deviations by selecting available CHP plants from a fixed, unsorted list and, if possible, adjusting their schedules to match the required balance capacity.

[4] They have, of course, to react to capacity requests from order agents.

Forecasting errors were modeled by distorting the thermal and electrical load curves of the simulated buildings via multiplication with gaussian distributed random numbers (mean 1.0, variance 0.05). For every season, the results were obtained by averaging five simulation runs of typical days with a temporal resolution of 15-minute intervals.

5.2 Simulation Results

Table 3 presents exemplary simulation results of three scenarios varying in N_{CHP}. The factor λ represents the percentaged improved compliance of the simulated VPP with it's initial operation schedule regarding a reference value $\Delta^{ref}_{(os,\overline{os})}$. The latter was obtained by simulating a scenario without consideration of reactive scheduling.

Table 3. Exemplary simulation results and key data for different seasons (S = summer, T = transitional season, W = winter)

N_{CHP} (P_{el})	season	$\Delta^{ref}_{(os,\overline{os})}$	$\Delta^{MARS}_{(os,\overline{os})}$	λ_{MARS}	$\Delta^{CDisp}_{(os,\overline{os})}$	λ_{CDisp}
11 (96.25 kW)	S	29.8 kW	25.5 kW	14.6 %	25.0 kW	16.2 %
	T	30.1 kW	27.1 kW	9.9 %	27.2 kW	9.5 %
	W	8.9 kW	8.2 kW	8.3 %	8.0 kW	10.0 %
50 (437.5 kW)	S	134.8 kW	115.2 kW	14.6 %	117.7 kW	12.7 %
	T	133.7 kW	121.6 kW	9.1 %	118.6 kW	11.3 %
	W	35.6 kW	32.4 kW	8.9 %	33.2 kW	6.6 %
100 (875 kW)	S	270.9 kW	223.6 kW	17.5 %	231.0 kW	14.7 %
	T	265.4 kW	234.3 kW	11.7 %	236.9 kW	10.7 %
	W	71.4 kW	66.4 kW	7.0 %	65.4 kW	8.4 %

Both MARS and CDisp perform best in summer. This effect is based on the comparatively low thermal demand, resulting in a higher flexibility of the CHP plants' operation times. While λ_{MARS} varies little in the first two scenarios and significantly improves in third scenario, λ_{CDisp} doesn't exhibit a distinct trend. A possible reason of the rather random performance is the following: CDisp acquires balance capacity by iterating over a fixed list of plants. As a consequence, the schedules of plants first in the list are changed often, thus leading to many subsequent schedule deviations. In contrast, MARS distributes the schedule adjustments over several CHP plants. Expressed by a significant improvement of λ_{MARS} in the third scenario, the results indicate a positive corellation of the number of DER in a VPP and the flexibility of reactive scheduling. In summary, the average improvements of the mean operation schedule deviation of about 10% to 15% are a good starting point for further investigations.

6 Conclusions and Future Work

The previous section indicated the suitability of a multiagent-based reactive scheduling system for virtual power plants. Improving MARS' performance

regarding the minimization of $\Delta_{(os,\overline{os})}$ is an important issue. In consideration of the results in Table 3, a promising approach is the development of heuristics for an optimized utilization of thermal storages. Thus, resource agents might be able to increase the reactive potential of VPP. As yet, the locating of negotiation partners for order agents is an unsolved problem (q.v. Sec. 4.2). Additionally, the communication overhead of the distributed control system disproportionally increases with the number of distributed energy resources in a VPP. In a static context, the introduction of hierarchically superior agents along the lines of PowerMatcher and DEZENT would be an appropriate solution of these problems. Yet, taking the domain-specific requirement DSR_3 from Sec. 4.1 into account, a static hierarchy is not adequate to deal with the inherent dynamics of future virtual power plants. Instead, implementing a dynamically adapting control hierarchy along the lines of holonic manufacturing systems [2] seems to be a more expedient approach. The potential of holonic virtual power plants will therefore be subject to further research.

References

1. Bitsch, R., Feldmann, W., Aumayr, G.: Virtuelle Kraftwerke - Einbindung dezentraler Energieerzeugungsanlagen (in German). etz 9, pp. 16-23 (2002)
2. Brussel, H.V., Wyns, J., Valckenaers, P., et al.: Reference Architecture for Holonic Manufacturing Systems: PROSA. Computers in Industry 37(3), 255–274 (1998)
3. Crastan, V.: Elektrische Energieversorgung, Teil 2 (in German). Springer, Heidelberg (2008)
4. Fischer, K., Müller, J.P., Pischel, M.: Cooperative Transportation Scheduling: an Application Domain for DAI. Applied Artificial Intelligence 10, 1–33 (1996)
5. Raheja, A.S., Subramaniam, V.: Reactive Recovery of Job Shop Schedules - A Review. International Journal of Advanced Manufacturing Technology 19, 756–763 (2002)
6. Kamphuis, R., Kok, K., Hommelberg, M., et al.: Massive coordination of dispersed generation using PowerMatcher based software agents. In: Proceedings of the 19th International Conference on Electricity Distribution, Vienna (2007)
7. Pielke, M., Tröschel, M., Kurrat, M., Appelrath, H.-J.: Operation strategies to integrate CHP micro units in domestic appliances into the public power supply. In: Proceedings of the VDE-Kongress 2008, Munich (2008)
8. Pudjianto, D., Ramsay, C., Strbac, G.: Virtual power plant and system integration of distributed energy resources. In: Renewable Power Generation IET(1), pp. 10–16 (2007)
9. Wedde, H.F., Lehnhoff, S., Handschin, E., et al.: Real-Time Multi-Agent Support for Decentralized Management of Electric Power. In: Proceeding of the 18th Euromicro Conference on Real-Time Systems (ECRTS 2006), pp. 43–51 (2006)

Concurrently Decomposable Constraint Systems

Cees Witteveen[1], Wiebe van der Hoek[2], and Nico Roos[3]

[1] Delft University of Technology, Dept of Software Technology,
P.O. Box 5031, 2600 GA Delft, The Netherlands
[2] University of Liverpool, Dept of Computer Science,
Ashton Street, Liverpool, L69 3BX, United Kingdom
[3] Maastricht University, Maastricht ICT Competence Center,
P.O. Box 616, 6200 MD Maastricht, The Netherlands

Abstract. In constraint satisfaction, decomposition is a common technique to split a problem in a number of parts in such a way that the global solution can be efficiently assembled from the solutions of the parts. In this paper, we study the decomposition problem from an autonomous agent perspective. Here, a constraint problem has to be solved by different agents each controlling a disjoint set of variables. Such a problem is called *concurrently decomposable* if each agent is *(i)* capable to solve its own part of the problem independently of the others, and *(ii)* the individual solutions always can be merged to a complete solution of the total problem. First of all, we investigate how difficult it is to decide whether or not a given constraint system and agent partitioning allows for such a concurrent decomposition. Secondly, we investigate how difficult it is to find suitable reformulations of the original constraint problem that allow for concurrent decomposition.

1 Introduction

Our problem is simple to state: Let C be a set of formulae (or constraints) over a partitioned set $X = \{X_i\}_{i=1}^n$ of variables. Each block X_i of variables is controlled by an actor A_i who, independently from the other actors, tries to find a satisfying assignment τ_i for its subset $C_i \subseteq C$ of formulas over X_i. Suppose that these *local assignments* τ_i are merged to a global assignment τ for X. Can we guarantee τ to be a satisfying assignment for C if we don't have any control over the choice of the locally satisfying assignments τ_i? And if not, how can we change C such that it satisfies this property?

As a simple example, suppose that Alice and Bob plan a party. Alice would like to invite Charles (c) or Diane (d) or both, while Bob, independently from Alice, wants to make an (inclusive) choice between Fred (f), Gerald (g) and Harald (h). It is also known, however, that inviting both Charles and Gerald will result in a disastrous party, hence, we would not like to have them both invited. So, let $C = \{c \vee d, f \vee g \vee h, \neg(c \wedge g)\}$ and let $X = \{c, d, f, g, h\}$ be partitioned into $X_1 = \{c, d\}$ and $X_2 = \{f, g, h\}$. Now Alice, controlling $X_1 = \{c, d\}$, chooses a satisfying assignment for $C_1 = \{c \vee d\}$, while Bob, controlling $X_2 = \{f, g, h\}$, takes a satisfying assignment for $C_2 = \{f \vee g \vee h\}$. Note that $\neg(c \wedge g)$ does not belong to C_1 or C_2, since it is not a formula over X_1 and neither over X_2. In this case, it is easy to see that we can't guarantee the existence of a globally satisfying assignment τ. For example, Alice might choose $\tau_1 = \{c = 1,\ d = 0\}$,

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 153–164, 2009.

while Bob might take $\tau_2 = \{f = 0,\ g = 1, h = 1\}$, but then the constraint $\neg(c \wedge g)$ is violated by the merge $\tau = \{c = 1, d = 0, f = 0, g = 1, h = 1\}$ of these assignments, implying that the party is over.

Some questions can be raised almost immediately. For example, it is clear that the composition of satisfying local assignments cannot always be guaranteed to be a satisfying global assignment, but how difficult is it to *decide* exactly whether such a guarantee can be given or not? What is the connection between this problem and deciding (in)consistency of the set of formulae C? Does it help if we already know some assignment satisfying C? These questions are all pertaining to the role of available *information* in answering this decision problem.

There is, however, also another, more constructive, way of looking to this problem as suggested in the general description of it above: Suppose that we have a *no-instance* of the problem, that is, an instance for which such a global satisfying assignment cannot always be guaranteed. Could we, by *changing* this instance, turn it into a yes-instance? For example, if, in the example above we could force Bob to choose between Fred and Harald, without having the possibility to choose Gerald, the existence of a globally satisfying assignment can be guaranteed. But of course, this also raises a lot of questions, like, which conditions to impose on the changed problem instance? Shouldn't we aim at a minimal change? But what is minimality in this respect? And how difficult is it to find an allowable change or a minimal change?

In this paper, we will only touch upon some of these questions. First, we will briefly discuss some related work and provide some general motivation for this research subject. Then we provide some formalisation by introducing a general framework for *distributed* constraint satisfaction problems and we define our problem in a more precise way. Next, we discuss the complexity of some associated decision problems and the problems associated with (minimally) changing the problem such that a global satisfying assignment can be guaranteed.

2 Motivation and Background

In constraint satisfaction, decomposition is a common technique to split a problem in a number of parts in such a way that the global solution can be efficiently assembled from the solutions of the parts. Most of these decomposition techniques that have been applied are *structurally* motivated, that is, the decomposition is performed by analyzing the structure of the set C of constraints. On the basis of the properties of C the original problem is split into a set of subproblems that can be solved in an easier way. The sets of variables associated with the resulting subproblems do not need to be completely disjunct, but do need to cover the global set X of variables. In general, the subproblems are easy to solve by guaranteeing that the resulting subproblems are *acyclic*: it is well-known that acyclic constraint solving problems can be solved efficiently, i.e. in polynomial time [1]. (Here, acyclicity refers to a property of the (hyper)graph underlying an instance of the constraint problem, see [3].)

There is an extensive set of literature [2,4,5,6,8,11] on this constraint decomposition problem with quite a number of different approaches to achieve suitable decompositions, like e.g. bi-connected components, (hyper)tree decomposition, hinge decomposition, query decomposition, tree clustering methods, and so on. A common aspect of

all these approaches, however, is that *(i)* the structure of the problem (i.e., the set of constraints) dictates the way in which the subproblems are generated and *(ii)* the subproblems generated do not have to be completely independently solvable, that is, in general, the decomposition will not allow the problems to be *concurrently* solvable.

In this paper we would like to study the decomposition problem from an autonomous agent perspective. We assume that a general (constraint) problem has to be solved by different parties (agents) that, each using their own approach, do want to solve their own part of the problem. This own part is determined by the capacities of the agents and not vice-versa. Moreover, we assume that each agent wants to solve its subproblem completely independently from the others. Such independent solving of subproblems, for example, might be required if the agents are not able (or not willing) to communicate during the problem solving process.

Hence, this problem differs in some respects from the problem solved by these structural decomposition methods. First of all, while in structural decomposition methods the partitioning (or covering) of the variables is a *result* of the decomposition technique and depends upon the structure of the constraint problem, using the autonomous agent perspective we are interested in decomposition methods that take a *given* partitioning of the variables into account.

Secondly, we require a *complete decomposition* of the original problem instance, that is, we would like to find a set of subproblems that can be solved *concurrently* and *independently* to obtain a complete solution to the original instance.

Note that this concurrent decomposition problem viewed from the agent perspective also can be viewed as a *mechanism design* problem (see e.g. [10]: we have to design a method that ensures that, whatever solution an individual agent proposes to its part of the problem, such a solution always can be used to compose a total solution. Hence, this mechanism should ensure that no agent is capable to prevent the construction of a global solution by the choice of its own partial solution.

3 Preliminaries

We consider (abstract) constraint systems $\mathcal{S} = (X, D, C)$ where X is a (finite) set of variables X, D is a set of (value) domains D^i for every variable $x_i \in X$ and C is a set of constraints on X.

We assume constraints $c \in C$ to be specified as formulas over some language. A solution s of the system is an assignment $s = \{x_i := d_i\}_{i=1}^n$ of all variables in X such that each $c \in C$ is satisfied. A *partial solution* is just an assignment to a subset $X' \subseteq X$ of the variables. We will assume that we have a constant d_i in the language for every domain element d_i, and we often will identify d_i with d_i and think of an assignment $s = \{x_i := \mathsf{d}_i\}_{i=1}^n$ as a *formula* $\bigwedge_{i=1}^n (x_i = \mathsf{d}_i)$ or a set of formulas $\{(x_i = \mathsf{d}_i)\}_{i=1}^n$. To preserve generality, we don't feel the need to specify the set of allowable operators used in the constraints $c \in C$ and their interpretation.

By $Sol(\mathcal{S})$ we denote the set of solutions s, i.e., satisfying assignments, to a constraint system $\mathcal{S}$. The system $\mathcal{S}$ is called *consistent* if $Sol(\mathcal{S}) \neq \emptyset$. For every $c \in C$, let $Var(c)$ denote the set of variables mentioned in c. For a set of constraints C, we put $Var(C) = \bigcup_{c \in C} Var(c)$. Given $\mathcal{S} = (X, D, C)$ we obviously require $Var(C) \subseteq X$.

Similarly, if D is a set of value domains D^i for variables $x_i \in X$ and we have $X' \subseteq X$, then $D_{X'}$ is the set of value domains $D^i_{X'}$ for variables $x_i \in X'$ with the obvious condition that for all $x_i \in X'$, $D^i = D^i_{X'}$. Given a set of constraints C and a set of variables X' we let $C_{X'}$ denote the subset $\{c \in C \mid Var(c) \subseteq X'\}$. Furthermore, if $X_1, X_2, \ldots, X_n$ are subsets of X and, for $i = 1, 2, \ldots, n$, s_i is an assignment of values to variables in X_i, then the *composition* $s = s_1 \sqcup s_2 \sqcup \ldots \sqcup s_n$ denotes an assignment of values to variables in $X_1 \cup X_2 \cup \ldots \cup X_n$. In particular, this assignment is well-defined if these sets X_i are pairwise disjoint.

3.1 Simple Constraint Systems

Given a constraint system $\mathcal{S} = (X, D, C)$ we often want to concentrate on the constraints relevant to a subset X' of the variables X. Selecting such a subset of variables and the constraints associated with it will induce just another constraint system, being a subsystem of the original system:

Definition 1. *Let $\mathcal{S}' = (X', D', C')$ and $\mathcal{S} = (X, D, C)$ be two constraint systems. Then we say that $\mathcal{S}'$ is a subsystem of $\mathcal{S}$, written $\mathcal{S}' \sqsubseteq \mathcal{S}$, if the following holds:*

1. $X' \subseteq X$
2. $D' = D_{X'}$
3. $C' = C_{X'}$

Furthermore, let $s = \{x_i := d_i\}_{i=1}^n$ be a solution for $\mathcal{S} \sqsupseteq \mathcal{S}'$. Then $s_{\mathcal{S}'} = s_{(X', D', C')}$ is the assignment $\{x_i := d_i \mid x_i := d_i \in s, x_i \in X'\}$.

It seems reasonable to assume that if a global constraint system $\mathcal{S} = (X, D, C)$ is consistent, any subsystem $\mathcal{S}' = (X', D_{X'}, C_{X'})$ with $X' \subseteq X$ derived from it, is also consistent (note that this hinges on monotonicity of the underlying logic).

Note that, by definition of a constraint system $\mathcal{S} = (X, D, C)$, we have $Var(C) \subseteq X$. Now, given a constraint system $\mathcal{S} = (X, D, C)$, there are at least three natural ways to obtain a subsystem $\mathcal{S}' = (X', D', C')$ from it:

1. Fix a set $X' \subseteq X$ and from that, derive D' and C' using Definition 1. In this case, we will write $\mathcal{S}' = \mathcal{S}_{X'}$.
2. Fix a subset set D' of value domains from D and find a set X' such that $D' = D_{X'}$. This can always be done by removing from X those variables that take a value in a domain in D but not in D'. The set of constraints C' is then also directly obtained: $C' = C_{X'}$. We write: $\mathcal{S}' = \mathcal{S}_{D'}$.
3. Fix a subset C' of the constraints of C and find a smallest set X' such that $C' \subseteq C_{X'}$. This can always be done by removing from X those variables that do occur in C, but not in C'. The set of domain values D' is then also directly obtained: $D' = D_{X'}$. We write: $\mathcal{S}' = \mathcal{S}_{C''}$.

With respect to the subsystem relation $\sqsubseteq$, we assume our constraint systems to satisfy the following *Preservation* property:

Preservation

Let $(X_1, D_1, C_1) = \mathcal{S}_1$ and $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$. Then $s \in Sol(\mathcal{S}_2)$ implies $s_{\mathcal{S}_1} \in Sol(\mathcal{S}_1)$.

Constraint systems that satisfy Preservation will be called *Simple Constraint Systems*.

3.2 Distributed Constraint Systems

Sometimes constraint systems $\mathcal{S}$ are distributed, that is, there is a set of actors A_i, each being able to make assignments or adding relations for/to a *subset* X_i of variables and these agents are collectively responsible for producing a global solution for $\mathcal{S}$. More specifically, if $\mathcal{S} = (X, D, C)$ is a constraint system and $X_i \subseteq X$ is the subset of variables controlled by agent A_i, then $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$ is the subsystem that has to be solved by agent A_i, where D_{X_i} is the set of domains for the variables in X_i, and C_{X_i} is as defined above.

If $\bigcup_{i=1}^{n} X_i = X$, while for $1 \leq i \neq j \leq n$, $X_i \cap X_j = \emptyset$, the collection $\{X_i\}_{i=1}^{n}$ constitutes a *partitioning* of X, and each X_i is called a *block* of $\{X_i\}_{i=1}^{n}$. If $\{X_i\}_{i=1}^{n}$ is a partitioning of X, we let $\mathcal{S} = (\{X_i\}_{i=1}^{n}, D, C)$ denote a *distributed constraint* system.

4 The Concurrent Decomposition Problem

In general, the *distributed constraint solving problem* can be simply stated as follows:

> Given a distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^{n}, D, C)$, is it always possible to find a global solution $s \in Sol(\mathcal{S})$ using arbitrary solutions $s_i \in Sol(\mathcal{S}_i)$ for its induced subsystems $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$?

While there are quite a few proposals for solving distributed systems [12], they almost all come down to some (distributed) backtracking process needed to resolve conflicts between partial solutions. Basically, what we are interested in are *backtracking-free* concurrent solutions. That is, we would like to investigate the following *concurrent decomposition* problem:

> Given a distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^{n}, D, C)$, is it true that the composition $s = s_1 \sqcup s_2 \sqcup \ldots \sqcup s_n$ of arbitrary solutions $s_i \in Sol(\mathcal{S}_i)$, where $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$, is always a solution for the total system $\mathcal{S}$?

If the answer is yes, we say that a distributed constraint system is *concurrently decomposable*. This has to be conceived as a decomposition that allows for concurrent solving of the decomposed parts of the system.

Definition 2 (Concurrent decomposition). *A (consistent) distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^{n}, D, C)$ is concurrently decomposable if $Sol(\mathcal{S}_1) \sqcup \ldots \sqcup Sol(\mathcal{S}_n) \subseteq Sol(\mathcal{S})$, that is, for every $(s_1, \ldots, s_n) \in Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_n)$ it holds that $s = s_1 \sqcup s_2 \ldots \sqcup s_n \in Sol(\mathcal{S})$. Here, for every $i = 1, \ldots, n$, $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$.*

We note that most constraint systems $\mathcal{S}$ will not allow us to simply decompose $\mathcal{S}$ into partial constraint systems $\mathcal{S}_i$ derived from $\mathcal{S}$, determine the solutions s_i to the partial systems and then just merge or compose these (partial) solutions to obtain the solution to the original system.

Example 1. Take a simple constraint system $\mathcal{S} = (X, D, C)$ where $X = \{x_1, x_2\}$ and is partitioned into $X_1 = \{x_1\}$ and $X_2 = \{x_2\}$, and $D^1 = D^2 = \mathbb{N}$. Let $C = \{x_1 \neq x_2\} \cup \{n_i < x_i \leq m_i : i = 1, 2\}$ for some given numbers $n_1 + n_2 + 5 < m_1 + m_2$. It is easy to see that the partial solutions s_1 for $\mathcal{S}_{\{x_1\}}$ and s_2 for $\mathcal{S}_{\{x_2\}}$ cannot always be joined to a global solution, since x_1 might be given the same value as x_2.

Example 2. Take a meeting scheduler, which has the aim to schedule two different meetings at a University in one and the same week. Meeting m_1 should be among faculty members of a specific Department, its Head, and its Dean, while meeting m_2 involves the Head of Department, the Dean and the Vice Chancellor. Moreover, we should keep in mind that no person can attend different meetings at the same time. Again, the partial solutions cannot always be joined to obtain a global solution.

Note that in both cases splitting the problem into several parts means that some constraints c such as $x_1 \neq x_2$ (Example 1) and the constraint that two meetings cannot overlap if there is a person that should attend both of them (Example 2) are not taken into account while solving the partial constraint systems individually. These constraints are the so-called *inter-block* constraints. Therefore, concurrent decomposability should also be viewed upon as a specification of a special relation between the set of *intra-block* constraints C_{X_i} and this set of *inter-block* constraints.

It is not difficult to show that, indeed, if the sets C_{X_i} of constraints covered by the partition blocks X_i together imply *all* constraints in C, that is also the inter-block constraints $c \in C$ such that $Var(c)$ is not contained in a single partition block, then the constraint system is concurrently decomposable:

Proposition 1. *Let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ be a consistent distributed constraint system and for $i = 1, \ldots, n$, let $\mathcal{S}_i = (X_i, D_{X_i}, C_{X_i})$. Then $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_n) \subseteq Sol(\mathcal{S})$ iff $\bigcup_{i=1}^n C_{X_i} \models C$.*

Proof (Sketch). Assume that $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_n) \subseteq Sol(\mathcal{S})$. Take an arbitrary s satisfying $\bigcup_{i=1}^n C_{X_i}$. Then s can be written as $s = s_1 \sqcup s_2 \sqcup \ldots \sqcup s_n$ where each s_i satisfies C_{X_i} and therefore, $s_i \in Sol(\mathcal{S}_i)$. By assumption, $s \in Sol(\mathcal{S})$. Therefore, s satisfies C.

Conversely, assume $\bigcup_{i=1}^n C_{X_i} \models C$. Then every solution s satisfying $\bigcup_{i=1}^n C_{X_i}$ will satisfy C. Each such a solution s can be written as $s = s_1 \sqcup s_2 \sqcup \ldots \sqcup s_n$ where each s_i satisfies C_{X_i}. Hence, $Sol(\mathcal{S}_1) \times \ldots \times Sol(\mathcal{S}_n) \subseteq Sol(\mathcal{S})$. □

The last proposition suggests that the problem whether a given distributed constraint system is concurrently decomposable or not is computationally closely related to deciding propositional logical consequence, which is coNP-complete. Indeed, as the next proposition shows, this is the case, even in the most simple distributed cases:

Proposition 2. *Let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ be a distributed constraint system. The problem to decide whether $\mathcal{S}$ is concurrently decomposable is a coNP-complete problem.*

Proof. Membership of coNP is easy: just guess a set of solutions $\{s_i\}_{i=1}^n$, where each s_i is a solution guessed for subsystem $\mathcal{S}_i$. Now check for each $i = 1, 2, \ldots, n$ whether $s_i \in Sol(\mathcal{S}_i)$ and then check the compatibility of these solutions. The violation of inter-block constraints c is easily verified using the composed global solution s.

Completeness follows using the following reduction from the coNP-complete LOGICAL CONSEQUENCE problem (Given a set of variables U, a set of clauses C over U and a clause c, is c implied by C?): Given an instance (U, C, c) of this problem, we consider the distributed constraint system

$$\mathcal{S} = (\{U, \{x\}\}, \{\{0,1\}^i\}_{i=1}^{|U|+1}, C \cup \{c'\} \cup \{\neg x\}),$$

where the set of variables is partitioned in the set U and the set $\{x\}$, and $c' = c \cup \{x\}$ is the clause c extended with the new atom $x \notin U$. Clearly, using this partitioning, $\mathcal{S}$ is decomposed into two systems $\mathcal{S}_U = (U, \{\{0,1\}^i\}_{i=1}^{|U|}, C)$, and $\mathcal{S}_{\{x\}} = (\{x\}, \{\{0,1\}\}, \{\neg x\})$.

Now, assume (U, C, c) to be a no-instance of the Logical Consequence problem, so let τ be any assignment verifying C, but falsifying c. Then τ is a solution for $\mathcal{S}_U$ and together with the only possible solution $\{x = 0\}$ for $\mathcal{S}_{\{x\}}$ it constitutes an assignment $\tau \sqcup \{x = 0\}$ that also falsifies $c' = c \cup \{x\}$. Hence, $\tau \sqcup \{x = 0\}$ is a certificate for non-decomposability of the distributed constraint instance $\mathcal{S}$. For the converse, assume that $\mathcal{S}$ is a non-decomposable instance. Then there are local assignments τ_1 and τ_2 such that τ_1 is a solution for $\mathcal{S}_U$ and τ_2 is a solution for $\mathcal{S}_{\{x\}}$, while $\tau_1 \sqcup \tau_2$ does not satisfy $\mathcal{S}$. Then, clearly, τ_1 must satisfy C, but does not satisfy c. Hence, (U, C, c) must be a no-instance of LOGICAL CONSEQUENCE. □

Note that this proof shows that this problem is already coNP-complete for the simplest possible distributed case where a partition contains only 2 blocks.

Remark 1. It is well-known that for general constraint systems finding a solution is NP-hard [3]. We therefore might ask whether having additional information about a satisfying assignment would help us in solving the decomposition problem. This turns out not be the case:

Proposition 3. *Let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ be a distributed constraint system and $s \in Sol(\mathcal{S})$ a satisfying assignment. Then the problem to decide whether $\mathcal{S}$ is concurrently decomposable is coNP-complete.*

Proof. Take a formula $\phi(x_1, x_2, \ldots x_n)$ over some alphabet $X = \{x_1, x_2, \ldots, x_n\}$. Without loss of generality we may assume that $n > 1$. Consider the constraint system $\mathcal{S} = (X \cup \{y\}, D, C)$ where $C = \{\phi(x_1, x_2, \ldots x_n) \vee y, y \vee \neg y\}$, D is a set of $\{0,1\}$ domains and the partitioning of X is $X = \{\{x_i\}_{i=1}^n, \{y\}\}$. Let s be an arbitrary assignment where $y = 1$. $\mathcal{S}$ is consistent and is concurrently decomposable exactly iff $\phi(x_1, x_2, \ldots x_n)$ is a tautology, the latter being a coNP-complete problem. □

It is easy to see that the same proof can be used to show that the availability of a partial solution that can be extended to a complete solution will not alleviate the difficulty of the decomposition problem.

Finally, note that if we are given *all* satisfying truth assignments s to the constraint solving problem, the concurrent composability problem can be seen to be polynomially solvable, but of course, this comes at the price of an exponential blow-up: we might be forced to take into account exponentially many assignments.

5 Finding Suitable Concurrently Decomposable Alternatives

As we have seen, the problem whether a distributed constraint problem is concurrently decomposable is an intractable problem (unless P=NP). But what happens if we could *change* a particular instance in such a way that it would become a yes-instance of the concurrent decomposition problem? Let us consider an example we discussed before:

Example 3. Take the system $\mathcal{S} = (X, D, C)$ where $X = \{x_1, x_2\}$ is partitioned into $X_1 = \{x_1\}$ and $X_2 = \{x_2\}$, and $D^1 = D^2 = \mathbb{N}$. Let $C = \{x_1 \neq x_2\} \cup \{n_i < x_i \leq m_i\}$ for some given numbers $n_1 + n_2 + 5 < m_1 + m_2$. The system is not concurrently decomposable, but if we add the constraints 'x_1 is odd' and 'x_2 is even' to the set of constraints, joining individual solutions will always deliver a global solution. Hence, this latter instance is a yes-instance of the concurrent decomposability problem. □

A first obvious restriction on the set of allowable changes of a given distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ would be to preserve the set of solutions of $\mathcal{S}$: For every resulting system $\mathcal{S}'$ it must hold that $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$. A second restriction we impose is that the sizes $|\mathcal{S}|$ and $|\mathcal{S}'|$ of $\mathcal{S}$ and $\mathcal{S}'$, respectively, are polynomially related, i.e., there should exist a polynomial p such that $|\mathcal{S}'| \leq p(|\mathcal{S}|)$. If these conditions do hold, we say that $\mathcal{S}'$ is a *suitable concurrently decomposable alternative* for $\mathcal{S}$.

Of course, first of all one would like to know whether such suitable alternatives always exist. The following proposition shows that for every consistent distributed constraint system there always exists a suitable concurrently decomposable alternative:

Proposition 4. *Given a simple distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$, there always exists a polynomially related concurrently decomposable distributed system $\mathcal{S}' = (\{X_i\}_{i=1}^n, D, C')$ such that $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$.*

Proof. Since $\mathcal{S}$ is consistent, there exists a solution $s = \{x_i = d_i\}_{i=1}^n \in Sol(\mathcal{S})$. We show that the system $\mathcal{S}' = (\{X_i\}_{i=1}^n, D, C \cup s)$ is concurrently decomposable and satisfies the solution preservation condition. For an arbitrary i, take the subsystem $\mathcal{S}'_i = (X_i, D_{X_i}, C_{X_i} \cup \{s_{X_i}\})$. By Preservation, $s_{X_i} \in Sol(\mathcal{S}'_i)$ and every solution $s' \neq s_{X_i}$ will violate at least one constraint $x_j = d_j$ occurring in s_{X_i}. Hence, for every $X_i \in \{X_i\}_{i=1}^n$, $Sol(\mathcal{S}'_i) = \{s_{X_i}\}$. Likewise, we have $Sol(\mathcal{S}') = \{s\}$. Therefore, $\mathcal{S}'$ is concurrently decomposable and $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$. □

Note that the proof of this proposition immediately shows that finding a solution to a simple constraint system $\mathcal{S}$ is at least as hard as finding a suitable concurrently decomposable alternative $\mathcal{S}'$ for a distributed variant of $\mathcal{S}$: Once we have found a solution to $\mathcal{S}$, we can use this solution to obtain a suitable concurrently decomposable alternative. In fact, is not difficult to see that both problems (finding a solution for a constraint system $\mathcal{S}$ and finding a decomposable alternative for the distributed variant of $\mathcal{S}$) are polynomially related.

Proposition 5. *Given a consistent constraint system $\mathcal{S} = (X, D, C)$, the problem to find a suitable concurrently decomposable alternative $\mathcal{S}' = (\{X_i\}_{i=1}^n, D, C')$ such that $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$ and such that $\mathcal{S}'$ is polynomially related to $\mathcal{S}$ and the problem to find an arbitrary solution to $\mathcal{S}$ are polynomially related.*

Proof. Given Proposition 4, it suffices to prove that a solution $s \in Sol(\mathcal{S})$ can be obtained if a polynomially related concurrently decomposable alternative $\mathcal{S}'$ has been found. So let $\mathcal{S}$ be a constraint system. Consider a partitioning $\{\{x_i\}\}_{i=1}^n$ of the set $X = \{x_1, x_2, \ldots, x_n\}$ and let $\mathcal{S}'$ be a suitable corresponding distributed concurrently solvable alternative of $\mathcal{S}$. By applying node consistency we can simply verify for each of the variables x_i a value using the constraints $C'_{\{x_i\}}$. Since $\mathcal{S}'$ is polynomially related to $\mathcal{S}$ each such a solution can be found in polynomial time. The composition of these partial solutions constitutes a global solution $s \in Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$ and can be found in polynomial time, too. □

6 Concurrently Decomposable Alternatives and Minimal Change

Finding an arbitrary suitable concurrently decomposable system might not always what we want. As we have seen in the previous section sometimes quite a lot of additional constraints might be added and sometimes the set of solutions of the original system might be seriously affected. In general, therefore, instead of adding an arbitrary set of constraints, we would like to apply the idea of *minimal change* to finding a suitable concurrently solvable alternative: how could we *minimally* change the original system S to a concurrently decomposable alternative S'. Applying this idea of minimal change, basically, there are two different approaches:

1. *maximize* the set of solutions $Sol(\mathcal{S}')$ such that the difference $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ is minimized;
2. minimize the amount of *constraint change* necessary to obtain the system $\mathcal{S}'$.

We could view the first approach as a *semantically* inspired approach, and the second as a *syntactical* approach. While the latter ensures that the syntactical difference between the two constraint systems is minimized, the first approach does not care which syntactical changes have to be applied but takes care for minimizing the loss of information associated with the transition to a decomposable system.

Here the bad news is: both the syntactical and the semantical approach give rise to intractable problems. To start with the latter approach, let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ be an instance of the distributed constraint problem. Following the semantical approach, we would like to obtain a distributed system $\mathcal{S}'$ such that

1. $\mathcal{S}' = (\{X_i\}_{i=1}^n, D, C')$;
2. $Sol(\mathcal{S}') \subseteq Sol(\mathcal{S})$ and $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ is minimal;
3. $\mathcal{S}'$ is fully decomposable, i.e., $Sol(\mathcal{S}') = Sol(\mathcal{S}'_1) \times \ldots \times Sol(\mathcal{S}'_n)$.

This problem, however, can be easily shown to be intractable, even if the set of solutions to the original system is of polynomial size and can be obtained in polynomial time:

Proposition 6. *Let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C) = (X, D, C)$ be a distributed constraint system. The problem to find a set of decomposed subsystems $\{\mathcal{S}'_i = (X_i, D_i, C'_i)\}_{i=1}^n$ such that* (i) $\bigtimes_{i=1}^n Sol(\mathcal{S}'_i) \subseteq Sol(\mathcal{S})$ *and* (ii) $\bigtimes_{i=1}^n Sol(\mathcal{S}'_i)$ *is a cardinality maximal subset of $Sol(\mathcal{S})$, is an NP-hard problem.*

Proof (Sketch). Consider the COMPLETE BIPARTITE SUBGRAPH problem: Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a positive integer K does there exists a complete subgraph of order K in G? This problem can be easily shown to be NP-complete by a reduction from the CLIQUE problem. Let $G = (V_1 \cup V_2, E)$ be an instance of the NP-hard MAXIMUM COMPLETE BIPARTITE SUBGRAPH problem. We create an instance of the minimal change problem as follows: Let $\mathcal{S} = (X_1, X_2, D, C)$ be a distributed constraint system where $X = \{x_1, x_2\}$ is partitioned as $X_1 = \{x_1\}$ and $X_2 = \{x_2\}$ with domains V_1 and V_2, respectively. Let C contain the constraint $(x_1, x_2) \in r_E$ iff $\{x_1, x_2\} \in E$.

Finding a suitable concurrently decomposable alternative $\mathcal{S}'$ for $\mathcal{S}$ that would minimize the difference $|Sol(\mathcal{S}) - Sol(\mathcal{S}')|$ implies that we have to find two subsystems $\mathcal{S}_1 = (\{x_1\}, \{V\}, C_1)$ and $\mathcal{S}_2 = (\{x_2\}, \{V\}, C_2)$, such that $C_1 \times C_2$ is a maximal subset of $r()$ (Here we represent the unary relations C_1 and C_2 just by their extension, i.e. subsets of V_1 and V_2). But that is of course equivalent to finding a maximum complete bipartite subgraph of G. □

Taking the *syntactical* approach, let us define a distributed constraint system $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ to be *k-decomposable* if a partial assignment to k variables in X already suffices to decompose $\mathcal{S}$ in independently solvable subsystems.[1] Here, we assume that after adding the k assignments to the variables, the original system of constraints is simplified by taking these assignments into account, i.e., replacing the variables by their values.

Note that, by definition, a 0-decomposable distributed constraint system is just a concurrently decomposable system. Also note that a consistent distributed constraint system is always n-decomposable: just use a solution $s \in Sol(\mathcal{S})$ and add s to the set of constraints C.

The general problem to decide whether or not a system is k-decomposable for some $k \geq 0$ turns out to be harder than just checking whether the system is concurrently decomposable:

Proposition 7. *Let $\mathcal{S} = (\{X_i\}_{i=1}^n, D, C)$ be a distributed constraint system, a partitioning of X and k a positive integer. The problem to decide whether $\mathcal{S}$ is k-decomposable is Σ_2^p-complete.*

Proof. To show that the problem is in Σ_2^p, given a constraint system $\mathcal{S}$ and a partitioning $\{X_i\}_{i=1}^n$ for X, guess a partial solution s for k-variables in X and add the constraints $x = s(x)$ for all $x \in dom(s)$ to C. Then, use a coNP-oracle to check 0-decomposability of the resulting constraint problem $\mathcal{S}'$.

To show that the problem is Σ_2^p-hard, we take the Σ_2^p-complete SUCCINCT SET COVER problem [9]: Given a collection $T = \{\phi_1, \phi_2, \ldots, \phi_m\}$ of 3-DNF formulae on a set Σ of variables and a positive integer k, is there a subset T' of T with $|T'| = k$ such that $\vdash \bigvee_{\phi \in T'} \phi$? The reduction from this problem is as follows: Let $(\Sigma, T = \{\phi_1, \phi_2, \ldots, \phi_m\}, k)$ be an instance of SUCCINCT SET COVER. Construct a distributed constraint system $\mathcal{S} = (\{X_1, X_2\}, D, C)$ where

[1] Of course, there are many other ways to specify the syntactical approach and to define other ways of constraint change, for example by restricting the domains of variables.

1. $X_1 = \Sigma$, $X_2 = \{x_1, x_2, \ldots, x_m\}\}$, and, for $j = 1, \ldots n$, $x_j \notin \Sigma$;
2. C contains two constraints $\Sigma_{\phi_i \in T}(\phi'_i + x_i) \geq 1$ and $\Sigma_{i=1}^{m} x_i = m - k$, where each ϕ'_i is obtained from ϕ_i by replacing $\vee$ by $+$ and $\wedge$ by $+$;
3. D a set of $\{0, 1\}$-domains for each of the variables occurring in $X = \Sigma \cup \{x_1, \ldots x_m\}$.

Suppose there is a subset $T' \subseteq T$ of size k such that $\vdash \bigvee_{\phi \in T'} \phi$. Then, for every $\phi_i \notin T'$, add a constraint $x_i = 1$ to C. Consider the two subsystems $\mathcal{S}_{X_1}$ and $\mathcal{S}_{X_2}$. Due to the presence of the variables x_i in C, the set of constraints C_{X_1} is empty. Hence, any assignment s_1 to the variables in Σ can be proposed as a solution $s \in Sol(\mathcal{S}_\Sigma)$. Considering an arbitrary assignment s_2 satisfying $\mathcal{S}_{X_2}$ (after addition of the set of constraints $\{x_i = 1 : \phi_i \notin T'\}$). There is exactly one such assignment: $s_2 = \{x_i = 1 : \phi_i \notin T'\} \cup \{\{x_i = 0 : \phi_i \in T'\}$. We will prove that the assignment $s = s_1 \sqcup s_2$ is a solution to $\mathcal{S}$.

This only requires to prove that s will satisfy $\Sigma_{\phi_i \in T}(\phi'_i + x_i) \geq 1$. This is easy to see, since $\vdash \bigvee_{\phi \in T'} \phi$ implies that $\Sigma_{\phi_i \in T'}(\phi'_i + x_i) \geq 1$ for every assignment s. Since $s(x_i) = 1$ for every $\phi_i \in T - T'$, s also satisfies $\Sigma_{\phi_i \in T - T'}(\phi'_i + x_i) \geq 1$. Hence, by adding $m - k$ assignment constraints, the constraint system becomes decomposable.

The converse is proven along the same lines. □

Note that the proof of this proposition again shows that Σ_2^p-completeness already holds for the simplest distributed case where we have a partition into two blocks.

7 Discussion

Although there remains a lot to investigate, we can draw three preliminary conclusions. First, testing whether a distributed constraint system is concurrently decomposable, being a coNP-complete problem, is likely to be intractable. Secondly, given a constraint problem $\mathcal{S}$, the problem to find a suitable alternative for a distributed variant of $\mathcal{S}$ is as hard as finding an arbitrary solution to $\mathcal{S}$. Thirdly, finding a syntactically minimal change alternative that is concurrent decomposable, is an even harder problem occurring at the second level of the polynomial hierarchy. The reason is that there are two sources of complexity: finding a minimal set of additional constraints and, secondly, testing whether the resulting system is concurrently decomposable.

Although from a computational perspective these results might seem to offer a somewhat disappointing view on the applicability of concurrently decomposable systems, they also clearly point out where to look for efficiently constructible decomposable systems. As we have shown, the problem of finding a suitable decomposable alternative of a system is as hard (neglecting polynomial differences) as finding solutions to a constraint system. Therefore, we should look at efficiently solvable constraint systems. And, indeed, there are already clear examples of efficiently decomposable systems: Simple Temporal Networks (STNs), being efficiently solvable constraint systems, for example, have been shown to be concurrently decomposable in an efficient way using a technique called Temporal Decoupling [7].

We envisage three further directions for this kind of research. First of all, we would like to investigate whether some of the complexity results obtained in the general case

still do hold if we restrict the class of allowable constraints. For example, what are the most complex distributed constraint systems for which finding minimally different concurrently solvable alternatives is tractable? What are the distinguishing features of such classes? Next, we would like to investigate what happens if in the case of minimal change we do not constrain the objects that can be added to partial assignments i.e., unary constraints, but also allow refinements of general constraints to be added. How does this e.g. influence the complexity of the k-decomposability?

Thirdly, in studying decomposability, we should also pay attention to other solution concepts than solution preserving decompositions. For example, in distributing constraints we could allow parties to express their preferences or resource constraints and to add them to the constraint system without bothering about composability of solutions, but only guaranteeing that whatever constraints are added, preservation of global consistency is preserved, whenever local consistency is preserved.

Acknowledgements

We would like to thank Léon Planken for fruitful discussions and suggestions.

References

1. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. Journal of the ACM 30(3), 479–513 (1983)
2. Cohen, D.A., Gyssens, M., Jeavons, P.: A unifying theory of structural decompostions for the constraint satisfaction problems. In: Complexity of Constraints. Dagstuhl Seminar Proceedings 06401 (2006)
3. Dechter, R.: Constraint Processing. Morgan Kaufmann Publishers, San Francisco (2003)
4. Dechter, R., Pearl, J.: Tree clustering for constraint networks. Artif. Intell. 38(3), 353–366 (1989)
5. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural csp decomposition methods. Artificial Intelligence 124, 2000 (1999)
6. Hsu, C.-W., Wah, B.W., Huang, R., Chen, Y.: Constraint partitioning for solving planning problems with trajectory constraints and goal preferences. In: IJCAI, pp. 1924–1929 (2007)
7. Hunsberger, L.: Algorithms for a temporal decoupling problem in multi-agent planning. In: AAAI 2002 (2002)
8. Naanaa, W.: A domain decomposition algorithm for constraint satisfaction. J. Exp. Algorithmics 13, 1.13–1.23 (2009)
9. Schaefer, M., Umans, C.: Completeness in the polynomial-time hierarchy: A compendium. SIGACT News 33(3), 32–49 (2002)
10. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. CUP, New York (2009)
11. Wah, B.W., Chen, Y.: Constraint partitioning in penalty formulations for solving temporal planning problems. Artificial Intelligence 170(3), 187–231 (2006)
12. Yokoo, M., Hirayama, K.: Distributed breakout algorithm for solving distributed constraint satisfaction problems. In: AAMAS1996, pp. 401–408 (1996)

SMIZE: A Spontaneous Ride-Sharing System for Individual Urban Transit

Xin Xing[1], Tobias Warden[1], Tom Nicolai[2], and Otthein Herzog[1]

[1] Center for Computing and Communication Technologies,
Universität Bremen, Germany
{xing,warden,herzog}@tzi.de
[2] Urban Team, Bremen, Germany
nicolai@urban-team.com

Abstract. We introduce a ride-sharing concept for short distance travel within metropolitan areas which is designed to handle spontaneous ride-sharing requests of prospective passengers with transport opportunities available on short call. The system has been designed as a multiagent system. We present a methodology to determine the feasibility of our ride-sharing approach for specific metropolitan areas and predefined operation requirements using multiagent-based simulation. Concrete experiments have been conducted for the city of Bremen, (Germany) in the FIPA-compliant multiagent-based simulation system PlaSMA.

1 Introduction

A continuous increase of traffic volume is among the major problems for traffic planning and management in urban and larger metropolitan areas. According to statistics from Eurostat [1, chap.10], the increasing demand in mobility is thereby mostly met with more individual passenger cars. As a consequence, existing traffic routes become over-strained which leads to regular traffic congestion and transport delays. At the same time however, passenger cars in urban areas are seldom used to their full capacity. From both a logistic as well as an ecological perspective, there is therefore potential for optimization. A wide range of approaches has been proposed to counter these development, for instance, an expansion of public mass transit, the establishment of car-free zones or congestion charges. We consider a further alternative, namely the installation of a ride-sharing system for individual urban transit. Compared to public transit, this approach retains the planning flexibility of individual transport, both with respect to the travel schedule and travel destination. It also offers the benefits of the utilization of passenger cars to people who do not have a vehicle of their own.

The concept of ride-sharing has recently seen an increase in popularity. Besides traditional stationary ride-sharing organizations and offices, numerous web-based ride-sharing systems have been established. These service platforms usually provide an electronic market place where car owners can offer ride-sharing

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 165–176, 2009.

opportunities for travels scheduled in the near future on the one hand, and passengers can search for interesting offers and negotiate terms of operation on the other hand. Restrictions in the usability of this class of ride-sharing systems are twofold: First, due to a strong focus on long-distance travel between different cities across a country, the transport granularity is too coarse for urban transit. Second, classical ride-sharing focusses on users, both drivers and passengers, that have a plan to go on a journey several hours or even days after the interaction with the respective service provider (e.g. placing an offer of a ride-sharing opportunity). As a consequence, *spontaneous* operation for users acting on short notice, i.e. in a time frame of up to an hour, may not be served adequately.

However, the aforementioned restrictions have been partially handled in the context of several research projects. In 1995, the University of Washington started the Seattle Smart Traveler project to research the concept of *"dynamic rideshare matching"* [2]. Volunteers recruited from students and staff of the university were asked to use a ride-sharing prototype accessible via a web-interface over the course of a year. During that period both the service adoption and the rate of successfully handled requests was analyzed statistically. The personal logistic system Corona (1998) was designed to offer ride-sharing matching services for car owners, in the run-up to the start of individual travels [3]. Corona was evaluated in simulation with a real street map. The conducted experiments were thereby focused on the potential reduction in globally traveled kilometers that could be achieved using the proposed ride-sharing concept and the question whether the flexibility of transport could be retained at the same time. In 2006, the work groups of Winter and Nittel developed an ad-hoc shared-ride trip planning system where drivers and passengers were represented as agents which live in a grid network and communicate with each other via radio-based messages [4]. The goal of this research was the identification of suitable communication strategies for agents in mobile ad-hoc network environments.

Although these research prototypes handle spontaneous ride-sharing requests for car owners and passengers, the relationship between the numbers of participating drivers and passengers with respect to success rate and acceptable travel time has not been investigated so far. Hence, based on a multiagent system (MAS) for spontaneous ride sharing, outlined in the next section, and by means of multiagent-based simulation (MABS), we focus on the following research question. Given (a) a particular metropolitan area as well as (b) specific estimates of the average number of car owners willing to offer ride-sharing opportunities and (c) the average number of residents interested in using the spontaneous ride-sharing service at each point in time for that area: Is the operation of the ride-sharing system feasible with regard to a certain quality of service measured in terms of travel time and the percentage of successful ride negotiations?

2 Spontaneous Ride-Sharing Concept

Embracing the paradigm of a multiagent-based application design, the *Smize* ride-sharing system comprises two classes of software agents. First, there is the class

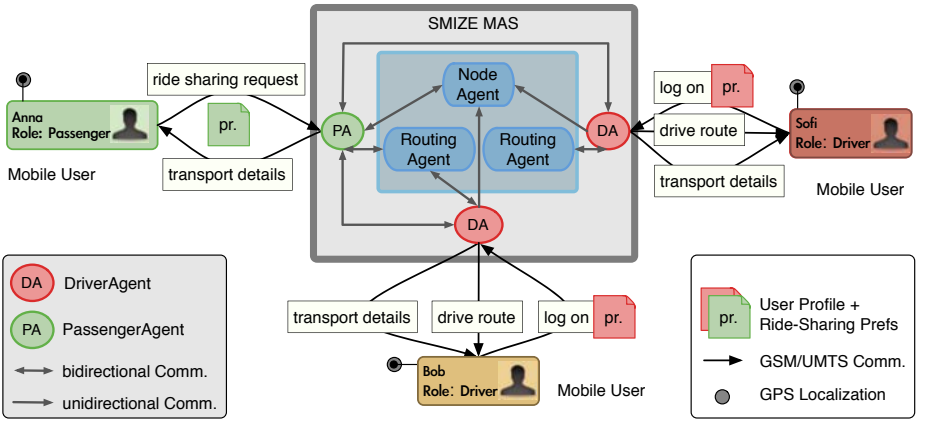


Fig. 1. *Smize* structure with scenario: Anna is a passenger who has a ride-sharing request; Bob and Sofi were identified as appropriate ride-sharing drivers for Anna

of user agents, which can be thought of as a software surrogate of ride-sharing users. Based on requirements specified by these users, their corresponding agents act proactively as goal-oriented decision makers with tactical autonomy in the sense of [5]. Driver agents (*DriverAgent*) advertise and manage new ride-sharing opportunities from drivers, while passenger agents (*PassengerAgent*) search and negotiate ride-sharing agreements for passengers looking for a transport opportunity. Neither of these tasks can be solved in isolation by the respective user agents. Problem solving in fact requires interaction with other members of the agent community. In particular, specific problem solving competences, such as the ability to compute routes are out-sourced to specialists acting as service agents. Instead of proactively and continuously pursuing their own agenda, these agents merely offer services for the user agents to help them perform their job. To be precise, the community of user agents relies on service of an appropriate number of routing agents (*RoutingAgent*) for the calculation of drive and walk routes and, for the time being, a single administrative agent (*NodeAgent*) which maintains and provides access to a database of drive route information of active vehicles as well as ride-sharing preferences of their drivers (cf. figure 1).

Routing agents in our system calculate the shortest drive route between the start and destination node of a vector street map. The $A^{\star}$-algorithm [6] is applied to determine the best way to the respective destination, using the driving time between two graph nodes for the heuristic function. The calculation of the driving time of a vehicle on a particular street segment thereby depends on the road classification of that segment. Relevant distinctions in metropolitan areas include amongst others residential areas (30 km/h), regular urban roads (50 km/h) or throughways with 60 km/h. Routing agents also offer a walk route service for passengers whose start or destination node is located in a pedestrian zone. In this situation, a boundary node of this pedestrian zone need to be identified. In succession, the shortest path from the start node of the passenger to the

Table 1. Personal information and ride-sharing preferences of passengers/drivers

(a) Personal User Information

User Profile	Passenger	Driver
Name	✓	✓
Gender	✓	✓
Smoker	✓	✓
#Passengers	✓	—
#Free Seats	—	✓
Phone Number	✓	✓

(b) Ride-Sharing Preferences

Ride-Sharing Preferences	Passenger	Driver
Gender	✓	✓
Smoker	✓	✓
Service Response Time	✓	—
Walk to Pickup Point?	✓	—
Transfer OK?	✓	—
Pickup Detour OK?	—	✓
Desired Travel Fees	✓	✓

boundary node, or from the boundary node to the destination can be computed. The passenger may then walk to the boundary node by foot along this walk route, in order to take a ride, or, if his or her destination is located in the pedestrian zone, he can get off his ride at the boundary node and walk the remainder of the route to the destination.

In the current version of our system concept, the *NodeAgent* manages a global database which comprises ride-sharing preferences (cf. table1) and sharing status (*free* or *occupied*) for each active driver, as well as a spatio-temporal representation of registered, i.e. currently active drive routes. These are stored as sequences of 3-tuples of the form (n, d, t_n) where n denotes a particular node of the street network, d the driver and t_n the corresponding expected arrival time of this driver at the node. Based on the temporally annotated route database, the *NodeAgent* can provide support for passenger agents in their search for appropriate transport candidates.

In the following, the operation of the user agents, the interaction between users and their corresponding user agents, as well as the system interaction between user agents and service agents are outlined step by step:

Once a driver has specified details of a new ride-sharing offer and communicated this offer along with his ride-sharing preferences via, for instance, a web front-end, *Smize* creates a new *DriverAgent*. This driver agent is responsible for (a) the acquisition of a detailed drive route through interaction with a routing agent, (b) the calculation of expected driving times on the route segments in order to annotate the route nodes with arrival times, (c) the forwarding of the temporally annotated route data to the *NodeAgent* and the user, and (d) the response to ride-sharing request of interested passenger agents. Initially, the sharing status of the driver is set to *free* rendering him eligible as potential ride-sharing provider.

For each passenger (or group of passengers traveling conjointly) communicating a ride request to our ride-sharing system, a new *PassengerAgent* is instantiated. This passenger agent is responsible for the identification of the driver candidate which can give the passenger a ride to the destination in minimal travel time. First, the passenger agent retrieves the *local neighborhood* of the passenger's starting point. It is defined as the subset of all street nodes which

can be reached via a path from the starting point whose total length is below a parametrized threshold, e.g. three kilometers. This set of nodes and the desired start time for the transport request are subsequently specified as parameters of a query to the *NodeAgent*. This service agent then computes the set of drivers to pass at least one street node within the local neighborhood of the passenger's starting point soon after the specified point in time. The set of pickup candidates and their respective ride-sharing preferences is sent back to the passenger agent which consequently filters candidates whose user profile and ride-sharing preferences do not match with those specified by the passenger.

If, at this point, the list of pickup candidates is empty, either because no suitable drivers have been suggested by the *NodeAgent* in the first place or all of the proposed candidates have been filtered out in the preferences matching process, the passenger agent adjourns the processing of its user's ride-sharing request for a specified amount of time (e.g. two minutes). Afterwards, it repeats the search process, based on the rationale that additional ride-sharing opportunities may have been offered after the processing of its initial query to the *NodeAgent*. The passenger agent may adjourn processing and consequently resume search for a fixed number iterations. If no ride-sharing candidates can be determined by then, the passenger is notified that the ride-sharing system is unable serve his particular ride-sharing request. Otherwise, i.e. when pickup candidates have eventually become available, the next step is the identification of those drivers that are also suitable transport candidates. Hence, the driving routes of the candidates are compared with the travel destination of the passenger. A driver is deemed suitable, if his drive route matches at least one of the street nodes in the local neighborhood of the passenger's destination. It is thereby accounted for that in order to drop of the passenger at his desired destination, the driver might have to drive a short detour.

In case there are no appropriate drivers that can take the passenger the whole way from start to destination, the passenger agent adapts its search strategy and begins to seek compound transfers with a single change of vehicle. An appropriate connecting driver must fulfill three conditions: (a) its drive route will intersect the route of one of the previously computed pickup candidates, (b) its drive route will pass through the passenger's local destination neighborhood, and (c) its ride-sharing preferences coincide with those of the passenger. If such a connecting driver can be identified, the group of drivers - one for pickup and the first part of the actual transport, another for the remaining transport - is included as a compound transport candidate. While conceptually possible, general multi-tier transports[1] has not yet been investigated further as it presupposes a careful investigation of the computational complexity when compiling routes with more than two transport tiers.

If the driver candidate list is non-empty at this point, the passenger agent selects the *best* driver (or the best combination of drivers, if there are no direct connections for the passenger). The pivotal criterion for selection is thereby the

[1] Which might become particularly interesting for instance when considering multi-modal transport combining individual and public transport.

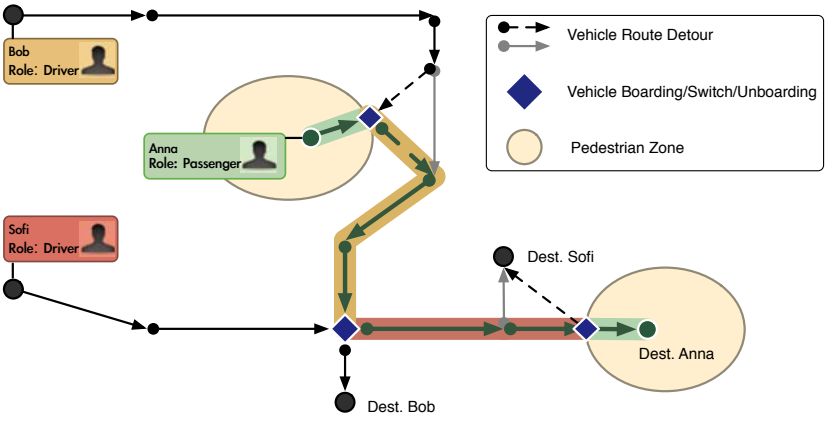


Fig. 2. Schema for a composite route offered as a transport solution

accumulated travel duration. Once the choice has been made, a ride-sharing request is sent to those driver agents representing the chosen driver candidate. The message contains the personal information of the passenger, the meeting point with the passenger and the drive route to get off point as determined by the passenger agent. Thereafter, the passenger agent sends back the complete transport details with driver(s) information, pick up and get off point(s) as well as the possible walk route to the passenger.

Upon reception of the ride-sharing request from a passenger agent, the notified driver agents change their sharing status to *occupied* which is retained until the transport is completed. We recognize, that this design decision neglects the capacity utilization of the transport vehicles and thus the possibility of additional passengers joining a transport which is already in progress.

Figure 2 presents a composite route of passenger Anna, her first driver Bob and connecting driver Sofi. In this case, both the start and destination point of Anna are located in a pedestrian zone. Therefore Anna first needs to walk to the pick up point on the boundary of the pedestrian zone where she can meet with Bob. In order to take up Anna, Bob has made a detour from his original route to the boundary point. Driver Sofi has waited at the crossing point with Bob for the arrival of Anna, and drives Anna to a suitable drop off point on the boundary of the pedestrian zone in which her destination is located. From there, Anna has to walk the final part of her travel, while Sofi continues to drive to her original destination along a new route. The total travel time of Anna consists of the service response time, the basic transport duration, the expected wait time for the arrival of the transport vehicles at the pick up points and walk times.

3 Multiagent-Based Simulation

Multiagent-based simulation (MABS) applies the concepts of multiagent systems to simulation. According to Herrler and Klügl [7, p. 575] MABS *"is a perfect*

means to represent and examine emergent effects in distributed systems. Multiagent simulation models may be used to gain insights into system interdependencies, to make predictions and also for testing software systems". Besides for function tests of our *Smize* prototype, we use MABS predominantly as a means to analyze on a conceptual level the feasibility of the particular spontaneous ride-sharing approach proposed in the previous section. That is, we use it do address the economically motivated question: If specific estimates about both the average number of car owners willing to offer a ride-sharing opportunity at each point in time and the number of residents interested in using the spontaneous ride-sharing service exist for a certain metropolitan area, is the operation of the ride-sharing system feasible with regard to a certain quality of service measured in terms of travel time and the percentage of successful ride negotiations?

For the conduction of our simulation experiments, we used the multiagent-based simulation environment PlaSMA[2] [8], which has been developed as flexible simulation platform for the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations" at the University of Bremen. PlaSMA is a simulation middleware which enhances the underlying FIPA[3] compliant agent platform JADE [9] with means for simulation. It handles experiment initialization, time management including message passing, as well as agent life-cycle management.

4 Experimental Setup

For the simulation of ride-sharing scenarios which differ in the numbers of participating passengers and drivers, we have implemented an external service agent as part of the simulation environment. It creates a continuous stream of user (i.e. passenger and driver) inquiries to the system which are consequently handled by new instances of adequate the user agent types introduced in section 2. These inquiries comprise the respective user profile, ride-sharing preferences as shown in the table on the left side of figure 3, as well as start and destination point which were chosen randomly from the street map shown on the right in figure 3. The generation process thereby respected the constraint that neither start nor end point of a *DriverAgent* can be located in a pedestrian zone. For all conducted experiments the *Smize* system could be realized with a basic set of service agents, namely a single instance of both of the *NodeAgent* and the *RouteAgent* without thus creating a noticeable performance bottleneck.

In order to run the simulation experiments on the street map of a real-word urban environment, detailed map material of the Bremen metropolitan area (11.000 crossings and 19.200 street segments) was downloaded from the OpenStreetMap[4] project. The map material was parsed into an ontologically modeled directed graph as required by the PlaSMA simulation system. Street segments

[2] Web site: `http://plasma.informatik.uni-bremen.de` (visited: July 17, 2009).
[3] *Foundation for Intelligent Physical Agents.* Web site: `http://www.fipa.org` (visited: July 17, 2009).
[4] Web site: `http://www.openstreetmap.com` (visited: July 17, 2009).

Ride-Sharing Preferences	Passenger	Driver
Gender	don't care	don't care
Smoker	don't care	don't care
Service Response Time	20 min.	
Walk to Pickup Point?	yes	
Transfer OK?	yes	
Pickup Detour OK?		yes
Desired Travel Fees	standard	standard

Fig. 3. Left: ride-sharing preferences for passenger and driver agents in the simulation experiments; Right: simplified street map of Bremen with 2.103 nodes and 4.448 edges

were thereby modeled as directed edges, crossings as nodes. Each directed edge was annotated with its speed limit and segment length. Since the ontology is imported into main memory (and validated in the process) upon simulation start, the load time for a scenario run strongly depends on the complexity of the directed scenario graph and the particular simulation system. In order to accelerate the load time, the original street map was simplified using the following strategies: (a) all of the streets in residential areas (30 km/h) were rejected; (b) streets with multi-segment curves, multiple traffic lanes or complex connections via traffic circles were represented with as few edges and nodes as possible. The resulting graph that has been used in all simulation experiments was comprised of 2.103 street nodes and 4.448 edges (cf. figure 3, right).

For the basic experiment outlined in the following section, the travel time of a passenger is determined directly by the speed limit of those street segments passed by the utilized ride-sharing vehicle(s). We thereby use the simplifying assumption that a continuous travel at maximum allowed speed is possible. Thus, the simulation model did not yet reflect traffic densities or congestions which pose an effective upper bound to effective travel speeds.

In some experiments outlined below, it was necessary to simulate passengers whose desired travel route roughly features a certain predetermined length (cf. section 5.1). In these cases, a random street node was chosen and set as starting point of the passenger. From this node and along an arbitrary direction, a route according to the shortest travel time was constructed incrementally until the travel time as computed by the $A^{\star}$-algorithm roughly corresponded with the required travel length. The end node of this route was set as the destination of this passenger.

5 Results

For the evaluation of the *Smize* concept, simulation experiments were conducted for the investigation of the relationship between travel time of passengers and number of participated drivers and in a further step for the comparison of the travel time with public transport.

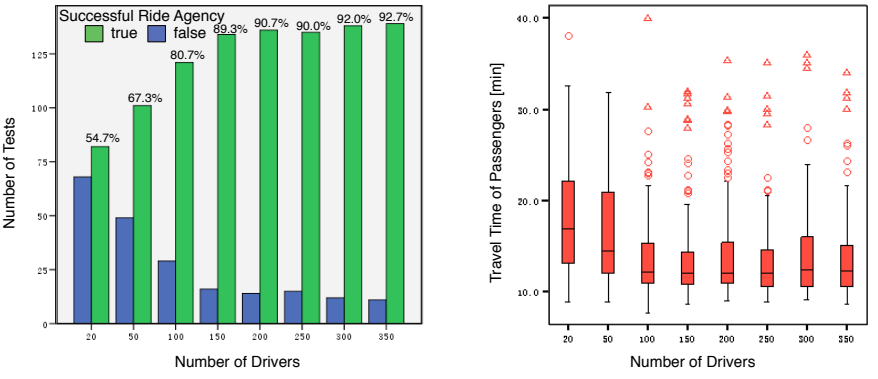


Fig. 4. The simulation result for one passenger with different number of participating drivers (with a maximum of 350). Left: The frequency of successful ride-sharing; Right: Distribution of travel time of a single passenger.

5.1 Influence of the Number of Smize Participants on Travel Time

The total number and distribution of active participants of a spontaneous ride-sharing system is a critical factor with regard to service quality, measured in the ratio of successful ride negotiations and the resulting travel times. We assumed (a) that the average travel time for certain routes is dependent on the ratio $r_{p/d} = |pass|/|driv|$ of passengers and active drivers, (b) that the desired success rate within a given travel time (i.e. *effectiveness* of operation) range requires at least $r_{p/d} \leq \theta_{succ}$, and finally (c) that the aspired transport *quality* requires an $r_{p/d} \leq \theta_{qual} \leq \theta_{succ}$.

In order to verify this assumption a two-tiered test scenario was set up. On the first tier of the test, the goal was to find the minimum number of active drivers $|driv|_{min}$ required for successful operation with a minimum number of ride-sharing requests. Thus, for each point in time there is only a single passenger ($|pass| = 1$) while the number of active drivers was kept constant for single scenario runs and was increased in discrete steps in succeeding scenario runs (i.e. $|driv| = n \in \{20, 50, 100, 150, 200, 250, 300, 350\}$). In each scenario run, 150 transport requests by the passenger were simulated. For each transport request the start and destination point of the passenger were drawn randomly from the vertices of the street map.

The travel length between the passenger's start and destination must be approximately equal each time such that a meaningful average travel time of the passenger can be computed. Long travel routes were avoided due to the fact that in a significant number of cases, the starting or destination point of the passenger was located near to the boundary of the map, where the throughput of vehicles was lower than in inner-city areas. This could cause unsuccessful ride-sharing cases, due to the limitation of the simulated regional scope rather than the *Smize* concept. The simulation result for one passenger with a travel length corresponding to eight minutes of pure travel time and a varying number of participating drivers is shown in figure 4.

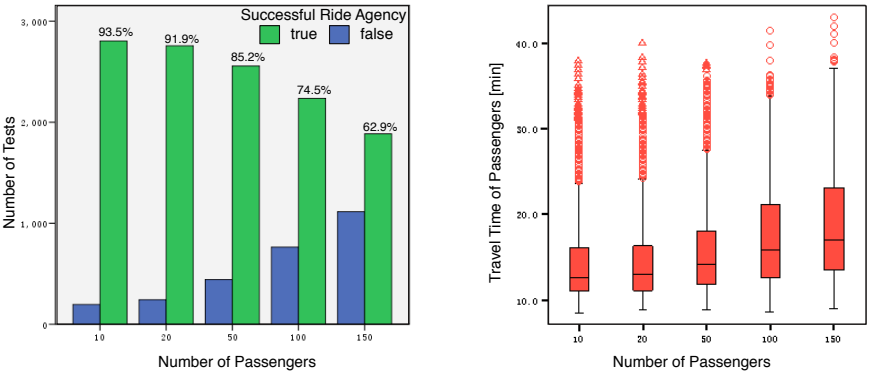


Fig. 5. The simulation result for various number of passengers (with a random travel length of 8 minutes) with 150 drivers. Left: The frequency of successful ride-sharing; Right: Distribution of travel time of passengers.

The result shows that with a constant amount of 150 participating drivers, the success rate of ride-sharing cases for a single passenger reaches 89.3%. A further increase of the amount of participating drivers yields only a low increment in success rate (only 3.4% when adding 200 active drivers). The average travel time of one passenger with 100 drivers is 13.8 ± 4.7 minutes and was not improved significantly by the increasing of the number of drivers. If we consider the results for achievable success rates together with the average travel time of a passenger, we can determine the minimal participating number of drivers for our spontaneous ride-sharing system in the Bremen scenario as $|driv|_{min} = 150$.

On the second tier of this test, we investigated for which amount of passengers the ride-sharing system could offer an acceptable service with the minimal participating number of 150 drivers. In this test, the number of drivers was kept fixed at $|driv|_{min} = 150$, while the number of passengers was increased in succeeding scenario runs. We started with 10 passengers sending ride-sharing requests to *Smize* concurrently. Each passenger has a random start and destination point. The travel length of each passenger is 8 minutes. This case was executed 300 times, so the total number of tests is 3000. Thereafter, the number of passengers was increased to 20 for each time and the executing times of this case was 150 ($20 \times 150 = 3000$). After this, the number of passengers was increased to 50, 100 and 150 and the executing times were respectively 60, 30 and 20.

Figure 5 presents the simulation result of this test. It shows that when there are more than 20 passengers sending a ride-sharing request in one time window ($r_{p/d} = 20/150$), the success rate of ride-sharing cases dropped below 90%. For the case that both of the participating number of passengers and drivers are 150 ($r_{p/d} = 1$), the success rate is still at 62.9%. The average value of travel time of passengers for the case with 10 passengers ($r_{p/d} = 10/150$) is 14.4 ± 5.1 minutes and for the case with 20 passengers 14.6 ± 5.1 minutes. Thus, there is no significant difference between those two cases. Both of them have only a

deviate of circa one minute in comparison with the case of a single passenger (see above) and are longer as the shortest travel time (8 minutes) up to 12 minutes.

To summarize, in the Bremen scenario the spontaneous ride-sharing system *Smize* can offer ride-sharing matching services for up to 20 passengers with a success rate of 90%, when there are at least 150 actively participating drivers in a same time window. If the travel segment of a passenger is not located in the boundary area where fewer resident live, the travel time with the ride-sharing service is expected to take up to 1.5 times the shortest travel time.

5.2 Comparison of Travel Time with Public Transport

A second test case was set up for the travel time comparison with public transport. Here, one passenger was simulated with different number of drivers ($|driv| = n \in \{20, 50, 100, 150, 200, 250, 300, 350\}$). The simulated average travel time of this passenger was compared with the travel time to be expended for the same travel segment using public transport. This test was executed three times. Each time, the start and destination point of the passenger were randomly chosen from the street map. The simulation result shows that with a minimal participating number of 50 to 200 active drivers, a single passenger can be taken to his or her destination with a high success rate of nearly 100%, if his or her start and destination point are analog to those of the test passenger. The travel time of the passenger with *Smize* (in three cases) is generally shorter than by taking public transport. The average reduction is circa the half. We have to note, however, that in this test case, there has been only a single passenger in this simulation who wants to take part in a ride-sharing. In the reality there are several passengers who send ride-sharing requests at the same time. As a consequence, the competition in the search for appropriate drivers might be significant. Therefore, we understand that our simulation result for this simplified test scenario underestimates the ride-sharing travel times to some degree.

6 Future Work

In its current incarnation the simulation environment employed as a testbed for the *Smize* system is based on a significantly simplified street map and uses a basic calculation of expected travel duration for vehicles on travelled route segments. Beneficial extensions towards greater realism of the simulation model include the implementation and connection of a realistic traffic model for the simulated metropolitan area or the use of real world traffic data. The latter could also act as a basis to render the simulation of user inquiries more realistic with regard to temporal and spatial variations in the demand of the ride-sharing service.

During the search for appropriate drivers for particular ride-sharing requests, the ride-sharing preferences of drivers and passengers were compared using rather strict matching rules which demand far-reaching agreement of the respective data. Thus, even the mismatch of a single preference item let to the filtering of the driver from the transport candidate list. In the future, it is possible to

order the transport candidates in a priority queue according to their respective degree of fulfillment of preference match and transport efficiency (expected driving and wait time for pick up). Then, even in situation where no completely matching candidate exists, slightly inferior yet tolerable alternatives might still be available.

In the experiment outlined in section 5.2, we opposed individual and public mass transport with regard to travel times when using exclusively either transport modality. Spontaneous ride-sharing was thus conceived as a potential substitute for public transit. Another approach is to take a rather holistic point of view an conceive ride-sharing and public transport as compliments to be combined in a multi-modal travel planning system.

Finally, we need to address the bottleneck induced by the use of a single *NodeAgent*, as it hinders the scaling of our current ride-sharing concept from several hundred to thousands of active participants.

Acknowledgement. The presented research was partially funded by the German Research Foundation (DFG) within the Collaborative Research Centre 637 "Autonomous Cooperating Logistic Processes: A Paradigm Shift and its Limitations" (SFB 637) at the University of Bremen, Germany.

References

1. Eurostat: Regions: Statistical Yearbook 2006. Technical report, European Communities, Luxembourg: Office for Official Publications of the European Communities (2006)
2. Dailey, D.J., Loseff, D., Meyers, D., Haselkorn, M.P.: Seattle Smart Traveler. In: Transportation Research Board 76th Annual Meeting, Washington DC, USA (1997)
3. Zegartowski, L.: Definition, Teilimplementation und Verifikation eines vollautomatischen Vermittlungssystems für den Personentransport. Ph.D thesis, Universität Bremen, Bremen, Germany (1998)
4. Winter, S., Nittel, S.: Ad-hoc Shared-ride Trip Planning by Mobile Geosensor Networks. Int. Journal of Geographical Information Science 20, 899–916 (2006)
5. Timm, I.J.: Strategic Management of Autonomous Software Systems. Professorial Dissertation, Universität Bremen (2006)
6. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn. Prentice Hall, Upper Saddle River (2003)
7. Herrler, R., Klügl, F.: Simulation. In: [10], pp. 575–596
8. Schuldt, A., Gehrke, J.D., Werner, S.: Designing a Simulation Middleware for FIPA Multiagent Systems. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, Sydney, Australia, pp. 109–113 (2008)
9. Bellifemine, F., Caire, G., Greenwood, D.: Developing Multi-agent Systems with JADE, 1st edn. Wiley Series in Agent Technologies. Wiley Inter-Science, Chichester (2007)
10. Kirn, S., Herzog, O., Lockemann, P., Staniol, O. (eds.): Multiagent Engineering: Theory and Applications in Enterprises. In: International Handbooks on Information Systems. Springer, Heidelberg (2006)

Towards a Verification Framework for Communicating Rational Agents

Nils Bulling[1] and Koen V. Hindriks[2]

[1] Department of Informatics
Clausthal University of Technology, Germany
bulling@in.tu-clausthal.de
[2] Faculty Electrical Engineering, Mathematics and Computer Science
Delft University of Technology, The Netherlands
k.v.hindriks@tudelft.nl

Abstract. We present an abstract framework for verifying communicative actions for rational agent programming languages. Firstly, a multi-agent verification logic based on the computational semantics is introduced; and subsequently, this multi-agent logic is embedded into a more expressive modal logic over a standard run-based semantics. We formally relate both logics, prove expressivity results, and argue why it is useful to have a (more expressive) *standard* modal logic and semantics at hand.

1 Introduction

In the literature the gap between agent programming languages and agent logics has frequently been discussed and first steps for bridging and analysing it have been done [5,6]. Just recently, a computational semantics for the GOAL agent programming language for communicative actions based on mental models was proposed in [4,1].

In this paper we relate such agent programming languages offering communication abilities to a "standardized" agent logic. For this purpose we propose an abstract setting for communicating agents, based on the message-passing system introduced in [3], and extend the verification logic from [5] to be applicable to the communicative setting. We continue to show that the computational semantics can be embedded into a run-based modal semantics. This result can be seen as a conservative extension of the single-agent result presented in [5] to the multi-agent setting introduced here; however, in this paper we leave out some operators whose addition is straightforward.

Due to the space limitations we will often refer to [1] for more details and the proofs of the main theorems.

2 Preliminaries

In the following we present the basic multi-agent model, sketch the essentials of an agent programming language and its extension by communicative actions.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 177–182, 2009.

The Multi-agent Model. For our multi-agent model we reuse the well-established theory on distributed systems from [3]. Our model assumes a fixed number of agents with associated *agent names* $\mathcal{A}gt = \{a_1, \ldots, a_n\}$. A *global state* g of a multi-agent system (MAS) is a tuple $\langle l_{a_1}, \ldots, l_{a_n}, l_e \rangle$ where l_{a_i} is the local state of agent a_i. We use g_a to denote the local state of agent a. The non-empty set $G = L_{a_1} \times \ldots \times L_{a_n}$ represents all *(global) states*.

In each state an agent a may perform an action, drawn from a set of *actions* Act_a where $Act_a \cap Act_b = \emptyset$ when $a \neq b$. We use α_i to denote actions and Act denotes the union of the action sets of all agents.

The effects of performing an action are represented by a *transition function* $\tau : G \times Act \rightarrow G$. Actions are assumed to update only the local state of the agent performing it. That is, $\tau(g, \alpha)_a = g_a$ whenever $\alpha \notin Act_a$. An exception to this rule will be made below for communicative actions. The behavior of a multi-agent system is given by a *run* r which is a mapping $\mathbb{N} \rightarrow G \times Act$. $r_1(i)$ (resp. $r_2(i)$) is used to denote the projection of $r(i)$ onto the first (resp. second) component of $r(i)$. We thus use an interleaving semantics to model the execution of a MAS, i.e. one action is executed per time step. A *multi-agent system model* $\mathcal{R}$, *system* for short, is defined as a set of runs.

Programming with Mental Models. Rational agents are programs that derive their choice of action from their beliefs and goals. An agent programming language provides a framework for programming with *mental models* that consist of an agent's beliefs and goals. Whereas in the single agent setting a mental model consists of the agent's own beliefs and goals only, in the multi-agent setting, that we consider here, we use the notion of a mental state that consists of mental models of other agents as well (cf. [4,1]). The idea is that these mental models are used to (partially) reconstruct the beliefs and goals of another agent.

The beliefs and goals of an agent are declarative sentences which are represented in standard propositional logic $\mathcal{L}_{PL}$ built over a set of propositional atoms $Atom$ and the usual Boolean connectives. $\models_{\text{PL}}$ denotes the usual consequence relation associated with $\mathcal{L}_{PL}$.

Formally, a *mental model* is a pair $\langle \Sigma, \Gamma \rangle$ with $\Sigma \subseteq \mathcal{L}_{PL}$ a *belief* and $\Gamma \subseteq \mathcal{L}_{PL}$ a *goal base* which satisfy the usual rationality constraints: (i),(ii) Consistency of beliefs and goals; and (iii) goals are not believed to be achieved (cf. [1,4]).

Finally, a *mental state* is a mapping m from $\mathcal{A}gt$ to mental models, i.e. $m(a) = \langle \Sigma, \Gamma \rangle$ is a mental model for each $a \in \mathcal{A}gt$. The set of all mental states is denoted by $MS(\mathcal{A}gt)$. The intuition is that a mental state m_a encodes a's beliefs about b's beliefs and goals by mapping agent name b to a mental model $m_a(b) = \langle \Sigma, \Gamma \rangle$ where Σ encodes b's beliefs and Γ encodes b's goals.

Agents need to be able to inspect their mental state and the different mental models part of it. Thus, the language of *mental state conditions* over $\mathcal{A}gt$, $\mathcal{L}_{MS}(\mathcal{A}gt)$, is defined by: $\psi ::= \mathbf{B}^a\phi \mid \mathbf{G}^a\phi \mid \neg\psi \mid \psi \wedge \psi$ where $a \in \mathcal{A}gt$ and $\phi \in \mathcal{L}_{PL}$. The semantics of mental state conditions is defined relative to a mental state m where $m(a) = \langle \Sigma_a, \Gamma_a \rangle$. So, we have, for instance, $m \models_{\text{MS}} \mathbf{B}^a\phi$ iff $\Sigma_a \models_{\text{PL}} \phi$; and $m \models_{\text{MS}} \mathbf{G}^a\phi$ iff $\exists\gamma \in \Gamma_a$ such that $\gamma \models_{\text{PL}} \phi$. The semantics for negation and conjunction is given in the usual way.

Communication Among Agents. The communicative actions that we introduce here affect the mental state of the *receiving* agent. Following [4,1], a communicative action is of the form $send(a, b, msg) \in Act_a$ where msg denotes a message that is being sent by agent a to b. Three indicators are introduced that intuitively correspond with the sentence types most often used in natural language: $\bullet$ for *declarative*, **?** for *interrogative*, and **!** for *imperative* sentences. Hence, a *message* is of the form $\bullet\phi$, $?\phi$, or $!\phi$ where $\phi \in \mathcal{L}_{PL}$.

3 The Formal Model for Communicating Agents

In this section we present a formal and abstract model for communicating rational agents based on the concepts introduced in Section 2 (again, we try to be as brief as possible and refer to [1] for a more detailed presentation). Mental states of an agent can be seen as concrete instantiations of the local states of Section 2; thus, we get $G = MS_{a_1} \times \ldots \times MS_{a_n}$ where MS_{a_i} denotes the set of mental states for agent a_i. The behavior of an agent is determined by its mental state. Here, the transition functions τ of Section 2 are therefore named *mental state transformers*. A corresponding run is called a(n) *(agent) trace*. $\tau(g, \alpha)_a(b)$ must satisfy the three rationality constraints of mental models for any $a, b \in \mathcal{A}gt$. Here, $\tau(g, \alpha)$ denotes a global state, $\tau(g, \alpha)_a$ denotes the mental state of agent a, and $\tau(g, \alpha)_a(b)$ denotes the mental model agent a associates with agent b, a notation we will often use below.

We extend a mental state transformer $\tau : G \times Act \to G$ to *message-passing mental state transformer* such that it can be applied to $send(a, b, msg)$ by imposing the following constraints:

1. If $b \neq a$, $\alpha \in Act_a$, $\alpha \neq send(a, b, \mathsf{m})$, then $\tau(g, \alpha)_b = g_b$
2. If $\alpha = send(a, b, \mathsf{m})$, then (i) $\tau(g, \alpha)_i = g_i \ \forall i \in \mathcal{A}gt \setminus \{b\}$,
   (ii) $\tau(g, \alpha)_b(i) = g_b(i) \ \forall i \in \mathcal{A}gt \setminus \{a\}$, and

$$\tau(g, \alpha)_b(a) := \begin{cases} \langle \Sigma_a \oplus \phi, \{\gamma \in \Gamma_a \mid \Sigma_a \oplus \phi \not\models \gamma\}\rangle & \text{if } \mathsf{m} = \bullet\phi \\ \langle \Sigma_a \ominus \phi, \Gamma_a\rangle & \text{if } \mathsf{m} = ?\phi \\ \langle \Sigma_a \ominus \phi, \Gamma_a \cup \{\phi\}\rangle & \text{if } \mathsf{m} = !\phi \end{cases}$$

Following [4,1], communicating a message m thus modifies the mental model $\langle \Sigma_a, \Gamma_a\rangle$ of the sender a maintained by receiver b. $\oplus$ and $\ominus$ are understood as update and revision operators. For details we refer to [4,1].

The Verification Language $\mathcal{L}_V$. The temporal language $\mathcal{L}_V$ to reason about communicating agents is an extension of the verification logic introduced in [5]. We enrich the logic by $\mathbf{B}_a^b\phi$ (a believes that b believes ϕ), and by $\mathbf{G}_a^b\phi$ (a believes that b has goal ϕ). The *verification language* $\mathcal{L}_V$ is given by the set of formulae χ defined by the following grammar:

$$\chi ::= \mathbf{B}_a^b\phi \mid \mathbf{G}_a^b\phi \mid \neg\chi \mid \chi \wedge \chi \mid \chi \mathbf{U} \chi \mid \mathbf{X}\chi \mid done_a(\alpha)$$

where $\phi \in \mathcal{L}_{PL}$, $\alpha \in Act_a$ and $a, b \in \mathcal{A}gt$. We also write $\mathbf{B}_a$ for $\mathbf{B}_a^a$ and $\mathbf{G}_a$ for $\mathbf{G}_a^a$. A trace generated by several agents and a message passing mental state

transformer serves as a model for $\mathcal{L}_V$. Given such a trace t and a time point $i \in \mathbb{N}$ the semantics of $\mathcal{L}_V$-formulae is defined in a straightforward way:

$$
\begin{array}{ll}
t,i \models_{\mathrm{V}} \mathbf{B}^b_a\phi & \text{iff } g_a \models_{\mathrm{MS}} \mathbf{B}^b\phi \text{ where } g = t_1(i)\\
t,i \models_{\mathrm{V}} \mathbf{G}^b_a\phi & \text{iff } g_a \models_{\mathrm{MS}} \mathbf{G}^b\phi \text{ where } g = t_1(i)\\
t,i \models_{\mathrm{V}} \mathbf{X}\chi & \text{iff } t,i+1 \models_{\mathrm{V}} \chi\\
t,i \models_{\mathrm{V}} \chi\mathbf{U}\chi' & \text{iff } \exists j \geq i : t,j \models_{\mathrm{V}} \chi' \text{ and } \forall k : i \leq k < j \Rightarrow t,k \models_{\mathrm{V}} \chi\\
t,i \models_{\mathrm{V}} \mathit{done}_a(\alpha) & \text{iff } i > 0 \text{ and } t_2(i-1) = \alpha
\end{array}
$$

and in the usual way for negation and conjunction. This logic allows to verify basic properties of a multi-agent system.

4 Embedding $\mathcal{L}_V$ in the Modal Logic $\mathcal{L}_M$

One disadvantage of $\mathcal{L}_V$ is that it is *non-standard* and not very expressive. In this section we introduce the modal logic $\mathcal{L}_M$ which is used to reason about runs. Then, we relate the verification logic $\mathcal{L}_V$ and its semantics to the modal logic $\mathcal{L}_M$ and present expressiveness results.

$\mathcal{L}_M$: Syntax and Semantics. The language $\mathcal{L}_M$ given by the grammar:

$$\varphi ::= \mathsf{p} \mid \neg\varphi \mid \varphi \wedge \varphi \mid B_a\varphi \mid G_a\varphi \mid \bigcirc\varphi \mid \varphi\mathcal{U}\psi \mid \mathit{Done}_a(\alpha)$$

is built over atoms $\mathsf{p} \in \mathit{Atom}$ and the temporal constructs $\bigcirc\varphi$ for φ holds in the next state, $\varphi\mathcal{U}\psi$ for φ holds until ψ holds, belief operators $B_a\varphi$ for $a \in \mathcal{A}gt$ believes φ, goal operators $G_a\varphi$ for a has goal φ, and $\mathit{Done}_a(\alpha)$ for a has performed $\alpha \in \mathit{Act}$.

The behavior of a MAS is modelled by a set of runs (cf. Section 2); thus, an $\mathcal{L}_M$-model $\mathfrak{M}$ is a tuple $\langle \mathcal{R}, \{\mathcal{B}_a \mid a \in \mathcal{A}gt\}, \{\mathcal{G}_a \mid a \in \mathcal{A}gt\}, V\rangle$ consisting of a set $\mathcal{R}$ of runs, serial belief and goal accessibility relations, one for each agent $\mathcal{B}_a, \mathcal{G}_a \subseteq \mathcal{R} \times \mathbb{N} \times \mathcal{R} \times \mathbb{N}$, and a valuation function $V : \mathcal{R} \times \mathbb{N} \to \mathcal{P}(\mathit{Atom})$which labels states with the facts true in it.

Formulae are interpreted over $\mathcal{L}_M$-models in the standard way (see e.g. [3]). We use $\mathfrak{M}, r, i \models \varphi$ to denote that φ is satisfied on r at time i in model $\mathfrak{M}$. Again, we skip the standard cases (see [1] for more details):

$$
\begin{array}{ll}
\mathfrak{M}, r, i \models B_a\varphi & \text{iff } \forall(r',i') \in \mathcal{B}_a(r,i) : \mathfrak{M}, r', i' \models \varphi\\
\mathfrak{M}, r, i \models G_a\varphi & \text{iff } \forall(r',i') \in \mathcal{G}_a(r,i) : \mathfrak{M}, r', i' \models \varphi\\
\mathfrak{M}, r, i \models \bigcirc\varphi & \text{iff } \mathfrak{M}, r, i+1 \models \varphi\\
\mathfrak{M}, r, i \models \varphi\mathcal{U}\psi & \text{iff } \exists j : j \geq i \text{ and } \mathfrak{M}, r, j \models \psi \text{ s.t.} \forall k : i \leq k < j \Rightarrow \mathfrak{M}, r, k \models \varphi\\
\mathfrak{M}, r, i \models \mathit{Done}_a(\alpha) & \text{iff } i > 0 \text{ and } r_2(i-1) = \alpha \in \mathit{Act}_a
\end{array}
$$

We define $X_a(r,i) = \{(r',i') \mid X_a(r,i,r',i')\}$ for $X \in \{\mathcal{B}, \mathcal{G}\}$ and, as usual, abbreviate $B_a\varphi \wedge \varphi$ as $K_a\varphi$.

Equivalence and Correspondence Results. We formally relate the logics $\mathcal{L}_V$ and $\mathcal{L}_M$ by embedding $\mathcal{L}_V$ into $\mathcal{L}_M$. We do so by introducing a translation tr from $\mathcal{L}_V$-formulae to $\mathcal{L}_M$-formulae defined as stated below:

$$tr(\mathbf{B}_a^b\phi) = \begin{cases} B_a B_b \phi & \text{if } a \neq b \\ B_a \phi & \text{if } a = b \end{cases}$$

$$tr(\mathbf{G}_a^b\phi) = \begin{cases} B_a G_b \Diamond \phi & \text{if } a \neq b \\ G_a \Diamond \phi & \text{if } a = b \end{cases}$$

$$\begin{aligned} tr(\neg\varphi) &= \neg tr(\varphi) \\ tr(\varphi \wedge \psi) &= tr(\varphi) \wedge tr(\psi) \\ tr(\mathbf{X}\varphi) &= \bigcirc tr(\varphi) \\ tr(\varphi \mathbf{U} \phi) &= tr(\varphi) \mathcal{U} tr(\psi) \\ tr(done_a(\alpha)) &= Done_a(\alpha) \end{aligned}$$

We show that this translation preserves truth which shows that the logic $\mathcal{L}_M$ can be used to reason about communicating agents instead of the non-standard $\mathcal{L}_V$. Our first result shows that $\mathcal{L}_M$ and its models are at least as expressive as $\mathcal{L}_V$ over traces; i.e., the modal logic can be used to reason about traces.[1]

Theorem 1. *Let t be a trace. Then there is an $\mathcal{L}_M$-model $\mathfrak{M} = \langle \mathcal{R}, \{\mathcal{B}_a \mid a \in \mathcal{A}gt\}, \{\mathcal{G}_a \mid a \in \mathcal{A}gt\}, V\rangle$ and a run $r^t \in \mathcal{R}$ such that for all $\varphi \in \mathcal{L}_V$ and $i \in \mathbb{N}$ we have: $t, i \models_V \varphi$ iff $\mathfrak{M}, r^t, i \models tr(\varphi)$.*

To obtain a correspondence result in the other direction, it is clear we need to impose some constraints on $\mathcal{L}_M$-models to ensure they model mental states and meet the rationality constraints of mental states and message passing mental state transformers. The consistency requirements for beliefs and goals are satisfied due to the seriality of the belief and goal relations. To match the third condition (goals are not believed to be achieved), we introduce the following postulate:

(R1) $\forall a, b \in \mathcal{A}gt:\ \mathcal{G}_a^b(r,i) \subseteq [\![\Diamond\varphi]\!]_{\mathfrak{M}} \Rightarrow \mathcal{B}_a^b(r,i) \not\subseteq [\![\varphi]\!]_{\mathfrak{M}}$

where $[\![\varphi]\!]_{\mathfrak{M}} := \{(r,i) \mid \mathfrak{M}, r, i \models \varphi\}$, the *denotation of* φ, consists of the points that satisfy φ and $\mathcal{B}_a^b(r,i) := (\mathcal{B}_b \circ \mathcal{B}_a)(r,i) = \{(r',i') \mid \exists (r'',i'') \in \mathcal{B}_a(r,i) : (r',i') \in \mathcal{B}_b(r'',i'')\}$. $\mathcal{G}_a^b := \mathcal{G}_b \circ \mathcal{B}_a$ is defined analogously. The subscript $\mathfrak{M}$ is omitted if clear from context.

In order to be able to match the communication semantics of message-passing mental state transformers, two additional postulates are required. Let r be a run and $X \in \{B, G\}$. The second postulate says that only the beliefs and goals of an action executing agent may change provided it is not a send action; and the third that only the mental state of the agent who receives the message is allowed to change in the prescribed way.

(R2) If $send(\cdot,\cdot,msg) \neq r_2(i) \in Act_a$ then for all $c, d \in \mathcal{A}gt$, $c \neq a$: $X_c(r,i) = X_c(r,i+1)$ and $X_c^d(r,i) = X_c^d(r,i+1)$

(R3) If $r_2(i) = send(a,b,msg)$ then for all $c, d \in \mathcal{A}gt$: $X_c(r,i) = X_c(r,i+1)$ and $X_c^d(r,i) = X_c^d(r,i+1)$ except if:
$msg = \bullet\varphi$ and φ consistent then $\mathcal{B}_b^a(r,i+1) \subseteq [\![\varphi]\!]$;
$msg = ?\varphi$ and φ no tautology then $\mathcal{B}_b^a(r,i+1) \not\subseteq [\![\varphi]\!]$;
$msg = !\varphi$ and φ no tautology then $\mathcal{B}_b^a(r,i+1) \not\subseteq [\![\varphi]\!]$ and $\mathcal{G}_b^a(r,i+1) \subseteq \mathcal{G}_b^a(r,i) \cap [\![\Diamond\varphi]\!]$.

[1] Proofs can be found in [1].

A run is called *trace-consistent* if it satisfies **(R1-3)**; and an $\mathcal{L}_M$-model is said to be *trace-consistent* if it contains at least one trace-consistent run.

Theorem 2. *Let $\mathfrak{M}$ be a trace-consistent $\mathcal{L}_M$-model. For each trace-consistent run r, all $\varphi \in \mathcal{L}_V$, and $i \in \mathbb{N}$: $\mathfrak{M}, r, i \models tr(\varphi)$ iff $t, i \models \varphi$.*

Benefits of the Modal Logic Approach. Why do we need two logics ($\mathcal{L}_V$ and $\mathcal{L}_M$) for the same purpose? An advantage of $\mathcal{L}_M$ is that it is more standard and thus better comparable to other logics, it is more expressive and allows to reuse existing results and tools (e.g. wrt. model checking). Hence, $\mathcal{L}_M$ seems especially suitable for the specification and verification of communication in MAS.

5 Conclusion and Related Work

We proposed first steps towards a theoretical model for *communicating rational agents*. We extended the verification logic from [5] to be applicable to the new setting and introduced a more expressive modal logic over standard models, based on [3], *to reason about communicating agents*. Links between both logics were established allowing to use the benefits of the "standard" modal logic.

The expressiveness of the logic to reason about communicating agents is limited compared to other logics that have been proposed [7,2], and remains an issue for future research, but an advantage of our approach is that it is based on the computational semantics introduced in [4]. Our work is very much related to [5]; actually, it can be seen as an extension of it. Here, however, we are even more general and relate agent programming languages to standard modal logic rather than Cohen and Levesque's Intention Logic [2].

References

1. Bulling, N., Hindriks, K.V.: Communicating Rational Agents: Semantics and Verification. Technical Report, Clausthal, Germany, Clausthal University of Technology (2009)
2. Cohen, P.R., Levesque, H.J.: Communicative actions for artificial agents. In: Proc. of the 1st Int. Conf. on Multi-agent Systems, ICMAS 1995 (1995)
3. Fagin, R., Halpern, J.Y., Moses, Y., Vardi, M.Y.: Reasoning about Knowledge. MIT, Cambridge (1995)
4. Hindriks, K.V., van Riemsdijk, M.B.: A Computational Semantics for Communicating Rational Agents Based on Mental Models. In: The 7th International Workshop on Programming Multiagent Systems, ProMAS 2009 (2009)
5. Hindriks, K.V., van der Hoek, W.: GOAL agents instantiate intention logic. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS (LNAI), vol. 5293, pp. 232–244. Springer, Heidelberg (2008)
6. Meyer, J.-J.C.: Our quest for the holy grail of agent verification. In: Olivetti, N. (ed.) TABLEAUX 2007. LNCS (LNAI), vol. 4548, pp. 2–9. Springer, Heidelberg (2007)
7. Singh, M.P.: Multiagent Systems: A Theoretical Framework for Intentions, Know-How, and Communications. Springer, Heidelberg (1994)

Designing Organized Multiagent Systems through MDPs*

Moser Fagundes, Roberto Centeno, Holger Billhardt, and Sascha Ossowski

Centre for Intelligent Information Technologies (CETINIA)
University Rey Juan Carlos, Madrid, Spain
{moser.fagundes,roberto.centeno,holger.billhardt,
sascha.ossowski}@urjc.es

Abstract. In this paper we present an approach to design an Organized Multiagent Systems (OMAS) for teamwork. We use a general formal model for OMAS that employs the notion of organizational mechanisms. The purpose of such mechanisms is influencing the behaviour of the agents towards more effectiveness with regard to some objectives. To achieve our goal we use Markov Decision Processes (MDPs) as a framework to design the organizational mechanisms. In order to illustrate our approach we use the medical emergencies domain where ambulances have to be selected in order to assist and transport patients to the hospitals.

1 Introduction

Ongoing research goals on Multiagent Systems (MAS) include the development of autonomous agents capable of reasoning and acting in open systems. Heterogeneous agents with different designs may join these open systems, and there is no guarantee that they will follow any behavior pattern, neither that they understand other entities. In this context the organizational concept can be of great value to provide support and regulate these MAS, ensuring the well-functioning of the whole system.

The paper [2] claims that the concept of organization is not restricted to the existence of some entity with a global purpose. Organizational structures may also exist (or emerge) as a mean to aid agents in their decision making processes in an uncertain environment. That work proposes a general formal framework for organizations, founded on the idea of *organizational mechanisms* that can be classified as *informative* or *regulative*. Informative mechanisms are well-suited for MAS where the agents have incomplete (and possibly inaccurate) knowledge about the environment, including other agents. These agents have to estimate and evaluate the expected utility of each possible action course (or of the actions it is aware of) to decide what actions to take next. From a micro level perspective any additional information may improve the agent's decision. From a macro perspective the informative mechanisms can be used to influence the agents' behavior without any adjustment on their autonomy. On the other hand the

* The present work has been partially funded by the Spanish Ministry of Science and Innovation, grants TIN2006-14630-C03-02 (FPI grants program) and CSD2007-00022 (CONSOLIDER-INGENIO 2010).

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 183–188, 2009.

regulative mechanisms adjust the agents' autonomy by changing their capabilities and actions' outcome. This hard enforcement aims to regulate the agents' behavior when particular environmental states take place.

In this paper we explore *organizational mechanisms* for efficient teamwork in an Organized Multiagent System (OMAS). To achieve our goal we design the organizational mechanisms with Markov Decision Processes (MDPs)[1]. Such Markovian models are well-known and have already been applied successfully in multiagent teaming [5]. To illustrate our approach we use the medical emergencies context where people request medical assistance for emergencies. The employment of advances on MAS to improve this type of medical service [4,3] is an area of significant potential since agent systems provide tools to simulate and evaluate different organizational models before applying changes on real-world systems.

This paper is organized as follows: in Section 2 we define Organized Multiagent Systems and detail the organizational mechanisms to be explored along the paper; Section 3 presents our design for an OMAS for medical emergencies done with MDPs; finally in Section 4 we draw the conclusion and future work.

2 Organized Multiagent Systems

According to [2] an OMAS is a tuple $\langle RA, A, \chi, \phi, x_0, \varphi, OM \rangle$ where: RA is a set of rational agents; A is a possibly infinite action space that includes all possible actions that can be performed in the system; χ is the environmental state space; $\phi: \chi \times A^{|RA|} \times \chi \rightarrow [0..1]$ is the transition probability distribution which describes how the environment evolves as a result of agents' joint actions; x_0 stands for the initial environmental state of the system; $\varphi: RA \times \chi \times A \rightarrow \{0, 1\}$ is the agents' capability function describing the actions the agents are able to perform in a given environmental state $\varphi(a,x,ac)=1$ ($\varphi(a,x,ac)=0$) means that an agent is able (not able) to perform action ac in the state x; OM is a non-empty set of organizational mechanisms.

The organizational mechanisms OM can be divided in two kinds: *informative* and *regulative*. An informative organizational mechanism is a function that given a partial description of an internal state of an agent and taking into account the partial view that the mechanism has of the current environmental state, provides information:

$$\Gamma: S' \times \chi' \rightarrow I$$

where:

- S' represents the set of possible partial descriptions of agents' internal states;
- χ' is the set of partial views of environmental states;
- I represents an information space.

The information provided may consist of a set of actions an agent can take but it is possibly not aware of, a recommendation of a particular action, or information about the consequences that a given action may have.

Regulative mechanisms produce modifications in the environment with the aim to improve the system's behavior from a macro level perspective. They can be divided in two types:

– An *incentive* mechanism Υ_{inc} for an OMAS is a function that given a description of the environmental state of OMAS produces changes in the transition probability distribution of OMAS:

$$\Upsilon_{inc}: \chi' \rightarrow [\chi \times A^{|RA|} \times \chi \rightarrow [0..1]]$$

– A *coercive* mechanism Υ_{coe} for an OMAS is a function that given a description of the environmental state of OMAS produces changes in the agent's capability function of OMAS:

$$\Upsilon_{coe}: \chi' \rightarrow [RA \times \chi \times A \rightarrow \{0, 1\}]$$

3 An OMAS Approach for Medical Emergencies

Different emergency centres may have different ways for handling their assistances, however they have common roles and procedures: centres receive help requests and send ambulances to assist the patients; if the ambulance crew is not able to provide the adequate treatment in situ, then the patient is transported to a hospital.

3.1 OMAS Specification

In our medical emergencies OMAS the rational agents can play one of the following roles: *ambulance*, *patient* or *hospital*. It is assumed that the system has p ambulances, q patients and r hospitals.

$$RA = \{ambulance_1, ..., ambulance_p, patient_1, ..., patient_q, hospital_1, ..., hospital_r\}$$

The action space A consists of: *move* for moving to an adjacent position; $assist_j$ when an ambulance assists the j^{th} patient; $admit_j$ for admitting the j^{th} patient in a hospital; $release_j$ for releasing the j^{th} patient; *wait* for waiting for medical assistance; and *skip* for doing nothing.

$$A = \{move, assist_j, admit_j, release_j, wait, skip\}$$

An environmental state space consists of a set of features $\chi = \Xi_1 \times ... \times \Xi_n$, where each Ξ_i, $1 \leq i \leq n$, corresponds to a single feature. Each feature Ξ_i can have a single value to be selected from a vector of possible values. The set of features to be specified in our MAS are decomposed in following subsets:

$$\chi = \chi_{ambulances} \times \chi_{patients} \times \chi_{hospitals}$$

The $\chi_{ambulances}$ is composed by the ambulances' state and position. Their state $\Xi_{as[i]}$ can assume the values *available*, what means the ambulance is not committed and consequently it is available for a new mission, or $patient_j$, what corresponds to $ambulance_i$ assisting $patient_j$. The ambulances' position $\Xi_{ap[i]}$ can assume the value that corresponds to the area the ambulance is located in. Within $\chi_{patients}$ the feature $\Xi_{pp[j]}$ corresponds to the $patient_j$'s position, while the feature $\Xi_{ps[j]}$ stands for $patient_j$'s state. The

patient's initial state is *healthy*, but when she gets *sick* she requests help and changes her state to *waiting*. When the ambulance begins the assistance the $patient_j$'s state $\Xi_{ps[j]}$ assumes the value *ambulance*. When the patient is hospitalized her state assumes the value *hospital* until she becomes *healed*. Finally, the $hospital_k$'s location is represented in the feature $\Xi_{hp[k]}$.

$\chi_{ambulances} = \Xi_{as[1]} \times ... \times \Xi_{as[p]} \times \Xi_{ap[1]} \times ... \times \Xi_{ap[p]}$
$\chi_{patients} = \Xi_{ps[1]} \times ... \times \Xi_{ps[q]} \times \Xi_{pp[1]} \times ... \times \Xi_{pp[q]}$
$\chi_{hospitals} = \Xi_{hp[1]} \times ... \times \Xi_{hp[r]}$

$\Xi_{as[i]} = \{available, patient_1, ..., patient_q\}$ where $1 \leq i \leq p$
$\Xi_{ap[i]} = \{area_1, ..., area_n\}$ where $1 \leq i \leq p$
$\Xi_{ps[j]} = \{healthy, sick, waiting, ambulance, hospital, healed\}$ where $1 \leq j \leq q$
$\Xi_{pp[j]} = \{area_1, ..., area_n\}$ where $1 \leq j \leq q$
$\Xi_{hp[k]} = \{area_1, ..., area_n\}$ where $1 \leq k \leq r$

The set of available actions for a rational agent in a particular environmental state is given by a capability function $\varphi: RA \times \chi \times A \rightarrow \{0, 1\}$. Table 1 summarizes the set of available actions (column *A*) for the agents (column *RA*) in particular environmental states (column χ). As an example, the action *move* is available for ambulances in all states of the world. The action $assist_j$ is available for an ambulance only in environmental states where the ambulance and $patient_j$ occupy the same physical location, the ambulance is available and the patient is waiting for medical assistance.

Table 1. Capabilities for agents involved in the medical emergencies domain

RA	*A*	χ
$ambulance_i$	*move*	ALL
	$assist_j$	$(\Xi_{ap[i]} = \Xi_{pp[j]}) \wedge (\Xi_{as[i]} = available) \wedge (\Xi_{ps[j]} = waiting)$
	$release_j$	$(\Xi_{as[i]} = patient_j) \wedge (\Xi_{ps[j]} = ambulance)$
$hospital_k$	$admit_j$	$(\Xi_{hp[k]} = \Xi_{pp[j]}) \wedge (\Xi_{ps[j]} = sick)$
	$release_j$	$\Xi_{ps[j]} = healed$
$patient_j$	*move*	$\Xi_{ps[j]} = healthy$
	wait	$\Xi_{ps[j]} = sick$
ALL	*skip*	ALL

The function φ defines the agents' capabilities for particular states of the world, but it does not provide information regarding how the environment evolves when an agent executes an action. Such environmental modifications are driven by the transition function $\phi: \chi \times A^{|RA|} \times \chi \rightarrow [0..1]$, which gives the probability distribution over states of the world that can take place through the execution of actions on the current state.

To specify the function ϕ for each $x \in \chi$ can be exhausting for a χ with a high number of features. In order to facilitate the specification of ϕ (qualitative and quantitatively) we use the following rules:

(r1) if $patient_j$ executes *move* then she will move to the adjacent position;
(r2) if $patient_j$ executes *wait* then $\Xi_{ps[j]}$=*waiting*.
(r3) if $ambulance_i$ executes *move* then she will move to the adjacent position; if $\Xi_{as[i]}$= $patient_j$ and $\Xi_{ps[j]}$=*ambulance* then $\Xi_{pp[j]}$ assumes the same value as $\Xi_{ap[i]}$ (if $ambulance_i$ is transporting $patient_j$ then their position have to be the same);
(r4) if n ambulances execute $assist_j$ (assist the same patient) at the same time then the chances of success for each one is $1/n$; if $ambulance_i$ is successful then $\Xi_{as[i]}$= $patient_j$ and $\Xi_{ps[j]}$=*ambulance*, otherwise $\Xi_{as[i]}$=*available*;
(r5) if $ambulance_i$ executes $release_j$ then $\Xi_{as[i]}$=*available* and $\Xi_{ps[j]}$=*sick*;
(r6) if $hospital_k$ executes $admit_j$ then $\Xi_{ps[j]}$=*hospital*;
(r7) if $hospital_k$ executes $release_j$ then $\Xi_{ps[j]}$=*healthy*;

These rules represent the knowledge agents have a priori about the world. In exception of (r4) the remaining rules are deterministic. However these rules can be easily modified in order to model the uncertainty on the agent's actions, e.g., the rules of the actions could be adjusted to have a probability p of success.

3.2 Organizational Mechanisms

As previously stated organizational mechanisms consist of processes for regulating the behavior of agents. In our case, we concentrate on the aim to assure rapid assistance for each possible patient. We assume that once a patient is assisted, the ambulance crew will provide the best possible treatment including a fast transport to a hospital if required. As organizational mechanisms we propose to use incentive and informative actions. We specify a reward function R_{OM} that motivates the agents in the OMAS to behave like a MDP. In our case, we consider a reward function R_{OM}: $\chi \times RA \rightarrow [0..1]$, that assigns rewards to agents in environmental states. By default, all rewards are zero.

In our application the organizational action space A_{OM} contains incentive mechanisms (aom1–2) and informative mechanisms (aom3–5):

(aom1) change the rewards for ambulances when assisting a patient.
(aom2) change the rewards for ambulances being located at different areas.
(aom3) inform each ambulance that $patient_j$ at $area_x$ is waiting for assistance and the reward it would receive for assisting the patient.
(aom4) inform each ambulance that $patient_j$ at $area_x$ has already been attended.
(aom5) inform each ambulance about the reward it would receive in each area.

The informative actions represented in (aom3) and (aom5) inform agents about the benefit their actions will have. (aom4) is intended to avoid unnecessary movement of ambulances. The incentive actions (aom1) aims to assure assistance and tends to avoid situations where multiple ambulances go for the same patient. Finally (aom2) intends to keep the city covered by giving incentives to the ambulances to move to uncovered areas. The reward function R_{OM} is changed with respect to (aom1) and (aom2) as follows:

(aom1) $R_{OM}((\Xi_{as[i]}=patient_j), ambulance_i)=r$, where r is proportional to the distance $ambulance_i$ had to $patient_j$ in the moment when $patient_j$ asked for help. (r is normalized to the interval [0.5..1], where r=1 for the nearest and r=0.5 for the farthest ambulance);

(aom2) $R_{OM}((\Xi_{ap[i]}=area_x) \wedge (\Xi_{as[i]}=available), ambulance_i)=r$, where r is inversely proportional to the density of ambulances in $area_x$ (r is normalized to the interval [0..0.5], where r=0.5 for the area with the lowest and r=0 for area with the highest density of ambulances);

4 Conclusion and Future Work

This paper presents an OMAS design for effective multiagent teamwork in the medical emergency domain. Our approach aims to bridge the gap between the abstract organizational model and the OMAS development by exploring MDPs. In this paper we focused on the organizational mechanisms, leaving the agents' design outside of our scope. We assume that agents are rational and, thus, tend to maximize their rewards.

By employing MDPs to build organizational mechanisms we highlight how they can be thought. Since we are dealing with autonomous agents, there is no guarantee that the usage of such mechanisms will bring the desired outcomes. Through *informative* mechanisms agents can be persuaded by receiving relevant information, but there is a chance that these agents do not take that information into account. By using *incentive* mechanisms it is possible to change actions outcomes, however there is no guarantee the agents will perceive such changes. Additionally, these incentives may not have an influence on the pursuit of the agents' intentions remember agents may not be aware of others' goals. Combining *incentive* with *informative* mechanisms, we assure that agents are aware of the possible outcomes their actions will have and can act accordingly.

Future work on Organized MAS is twofold. The first research direction consists of applying learning techniques for MDPs to design dynamic organizations. The second one aims the investigation of medical emergency scenarios with complex interactions, competition and establishment of commitment between involved parties (e.g. ambulance companies have to compete to assist a patient, and independent of the circumstance at least one ambulance has to be assigned to help that patient).

References

1. Bellman, R.: A Markovian Decision Process. Journal of Mathematics and Mechanics 6, 679–684 (1957)
2. Centeno, R., Billhardt, H., Hermoso, R., Ossowski, S.: Organising MAS: A Formal Model Based on Organisational Mechanisms. In: Shin, S.Y., Ossowski, S. (eds.) 24th Annual ACM Symposium on Applied Computing (SAC 2009), Hawaii, USA, pp. 740–746. ACM, New York (2009)
3. Centeno, R., Fagundes, M.S., Billhardt, H., Ossowski, S.: Supporting Medical Emergencies by MAS. In: Håkanson, A., Nguyen, N.T., Hartung, R.L., Howlett, R.J., Jain, L.C. (eds.) KES AMSTA 2009. LNCS, vol. 5559, pp. 823–833. Springer, Heidelberg (2009)
4. López, B., Innocenti, B., Busquets, D.: A Multiagent System to Support Ambulance Coordination of Urgent Medical Transportation. IEEE Intelligent Systems 23(5), 50–57 (2008)
5. Nair, R., Tambe, M.: Hybrid BDI-POMDP Framework for Multiagent Teaming. Journal of Artificial Intelligence Research (JAIR) 23, 367–420 (2005)

A Reference Architecture for Modelling of Emotional Agent Systems

Julia Fix and Daniel Moldt

University of Hamburg, Department of Informatics,
Vogt-Koelln-Str. 30, D-22527 Hamburg, Germany
http://www.informatik.uni-hamburg.de/TGI/

Abstract. In this contribution E-Mulan (Emotional Mulan (Multi-agent Petri nets)) are introduced. On the one hand it is a reference architecture for modeling emotion in MAS. On the other hand it is the technical realisation based on our tool set Renew and Capa.

1 Introduction

First efforts of the DAI research focused on the possibility of implementing the functionality of natural emotional phenomena in (multi-)agent systems, addressing problems of action selection, resource allocation, multi-agent co-ordination, social control and structuration of artificial societies etc (see e.g. [15] for an overview).

On the most general level, we aim to provide more profound theoretical underpinning for MAS-models of emotion. We expect to do so by modelling and integrating interdisciplinary approaches to emotion within a single modelling framework. For the development, analysis and understanding of socio-technical aspects of MAS we aim at analytical separation of cognitive and emotional components of a MAS model. This separation is illustrated in this paper with the proposed reference model of emotion based MAS architecture.

2 Conceptual and Technical Background

We seek to accomplish the integration of cognitive science and social science approaches to emotion using a general modelling technique that allows building a unique integrative model of emotion that (a) is usable in an MAS context and (b) provides general theoretical underpinnings for an implementation of emotion in (distributed) artificial systems.

2.1 Formal Modelling Techniques: Petri Nets and Reference Nets

A practical evaluation of the reference net formalisms has revealed its important advantages for modelling different sociological theories (see [10] and [11] for general compact summaries of the research in this area for more than a decade.).

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 189–194, 2009.

Basing upon these results, we propose the application of Petri Net / Reference Net modelling technique for modelling emotion-based processes and concepts in multi-agent systems.

The graphical representation of Petri nets, which is intuitive and easy to understand also without extensive additional instructions, has proven to be especially useful in interdisciplinary endeavours, constituting a general communications language, e.g. for collaboration of computer scientists and sociologists (see [10]; [7]). Unlike other automata formalisms Petri nets allow for direct modelling of concurrency (i.e. independent events and processes). Moreover, high-level Petri nets permit the representation of recursive structures and emergent processes and are able to inherently express structural as well as process concepts at the same time. Well established extensions of the basic formalism and sophisticated tool sets for most of these extensions exist.

Together with these extensions, Petri nets thus provide a powerful instrumentarium for modelling dynamic, hierarchical and recursive structures, as described by psychological, neurological and social theories of emotion. Due to its operational semantics and the broad range of available analytical methods, modelling with Petri nets allows an evaluation and formal checking of semantics of the models, however, neglecting the Java code. In addition, the formal representation of emotion models facilitates their integration into the computational domain. Further details on reference nets modelling are omitted here, but can be found in several of the given references in this text. The practical side of reference nets can be found in an efficient editing and simulating tool called Renew (Reference Net Workshop, [12]), which provides a powerful support for designing and evaluating Petri net / reference net models.

2.2 Social Models Based on the Mulan-Architecture

The proposed architecture is based on the Mulan multi-agent architecture ([6]; [13]), which is a conceptual implementation of the agent concept on the basis of the reference net formalism [8]. It has the general structure as depicted in the central section of figure 1.

While Mulan provides four conceptual levels of abstraction, that are used to describe hierarchies in a multi-agent system, Capa provides a FIPA-compliant embedding for practical use (FIPA, [4]). Since we concentrate on the underlying conceptual model, the implementation details are omitted here and can be found in [13]. While Mulan and Capa are usually used to directly model and / or implement usual distributed software artefacts, we use it here in the spirit of our Socionics projects (as documented in [10], [7], [11] etc.). This means that different levels of abstraction of software are associated with specific levels of social systems.

Each (white) box describes one level of abstraction in terms of the net hierarchy. Each upper level net contains net tokens, whose structures are made visible by the ZOOM lines.

The figure shows a simplified version of MULAN, since for example several inscriptions and all synchronous channels are omitted. Nevertheless MULAN is an executable model.

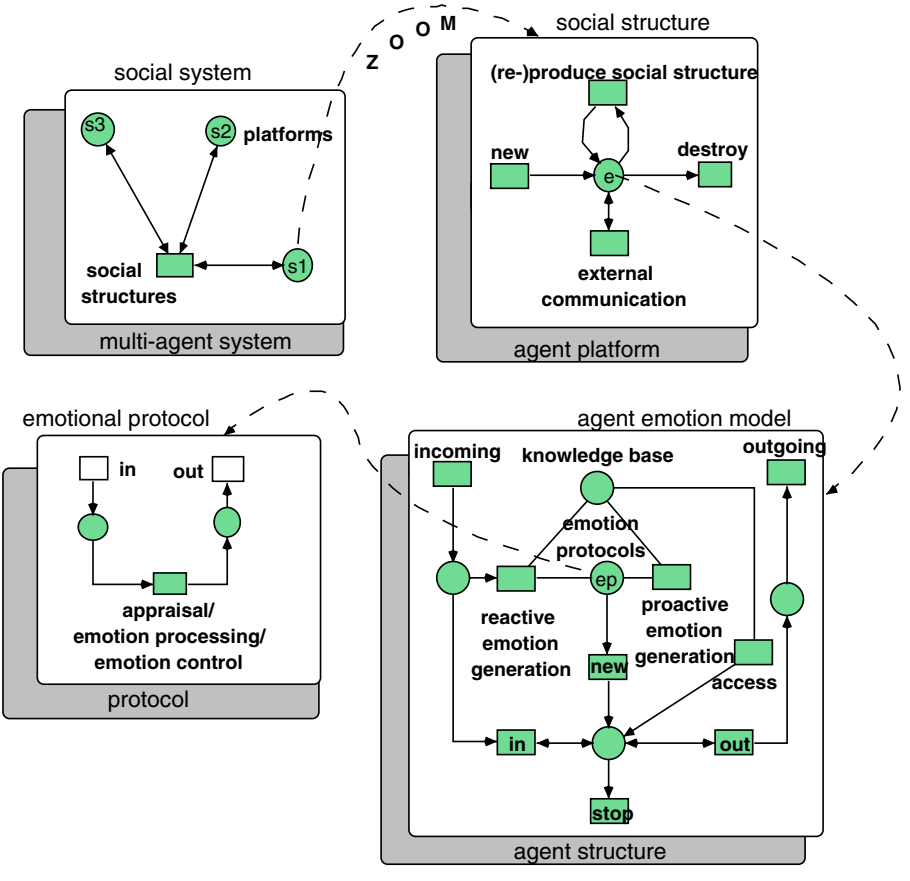


Fig. 1. A MULAN-based Reference Architecture for Modelling Emotion in MAS

2.3 Modelling Emotion within Mulan: E-Mulan

In this subsection we introduce a layered framework for modelling emotions, based on MULAN-concepts, that functions in parallel to the conventional non-emotional MULAN-architecture. The formal model, on which the representation bases, is the recursive formalism of reference nets, as described in the previous subsection.

General Setting. Assuming an important influence of emotion on fundamental processes of human and artificial intelligence (e.g. [9]), we aim to establish a modelling framework, which explicitly separates emotions from cognitive processes. In this way we expect to facilitate the investigation and implementation

of the interdependences between emotional and cognitive processes. We postulate that supplementary to cognitive parts of a model emotional counterparts exist. Actually within our previous models of sociological theories (see e.g. [10]) we integrated cognitive and emotional models, since this is a usual perspective in sociology: The mutual dependencies of emotion and cognition lead to an integrated modelling. We propose here an analytical separation of concerns for emotion and cognition.

We use the existing MULAN-architecture to model the cognitive component of the architecture and then enhance it with special components for modelling emotions, which is for its turn based on MULAN modelling concepts. The linkage between the proposed framework and the conventional architecture may be realized in a simple and homogenous manner, based on the applied modelling techniques: The existing (cognitive) models are completed with further models of emotion. The statements of the sociological (and other) emotion theories either deal with pure emotion-based interdependences, in which case they are to be integrated only in the emotion models, or emotions are additionally claimed to influence social structures and processes. In the latter case, linkage to the rational elements of the model has to be created, which is technically accomplished by means of synchronous channels. Thus any possible linkage of actions from different layers of the proposed architecture can be represented.

Based on our experiences from socionics projects (i.e. [10]; [7]), we distinguish four layers for modelling emotion as follows (these four layers can be found as the white boxes in Fig. 1):

Social System View. The first layer represents social aggregates (groups, fields etc.) that constitute the social structure of the (artificial or natural) society. On this layer social macro-structures, as they are specified in sociological theories of emotion, can be positioned.

Social Structure View. The second layer of the proposed framework specifies the social structures. These social objectives are defined through the platform and are valid for all actors/agents residing on the platform. They influence emotions of the individual and are influenced by emotions on their turn. Further, agents can interact and communicate their emotions by message exchange. At this level we can model theories of emotion, which describe the functions of emotion for the structuration of society or the interdependences between emotion and social control structures, like social norms, behavioural rules, etc. E.g., the social constructionist theories of emotion, which consider emotions as socially constructed phenomena, whereby the emotional arousal and expression is facilitated mainly by socially negotiated norms (feeling rules and expression rules [5] or display rules [2]) would be positioned at this layer.

Agent View. On the third layer we can describe the model of emotion of an actor or agent as a whole. The conventional (cognitive) model of an agent must be extended with an internal emotional model, specifying mechanisms of emotion generation (appraisal processes) by means of *emotional protocols*. In case of an emotional agent, not only activities of the agent, but also his emotions

are modelled with protocols. In accordance with the conventional MULAN agent model the selection of these protocols can succeed either reactively, corresponding to automatic emotional responses (primery or basic emotions [1]) that are generated as immediate reactions on certain (critical) events in the agent's dynamic environment, or proactively, by means of a complex deliberative process of emotion generation (appraisal), which can be started also without external eliciting condition and are specified through the relation of agents beliefs, desires and observations of the environment.

Through the parallel linking to a cognitive agent model we can specify the influence of generated emotion on deliberation, i.e. behaviour, action selection, planning etc., as far as these influences are specified in the modelled theory of emotion.

Emotion View. The last layer enables the representation of actually active emotions that were dynamically generated on the third level or that make up the basic underlying processes of the emotions. These emotions are structurally and conceptually equivalent to the dynamically generated behaviours/actions of an agent. They can run independent of each other and interact with each other via the agent, that is the environment for the emotions. Thus the proposed architecture enables modelling of interactions between agent's emotions. Futhermore, as protocols representing agents activities and emotions can run simultaneously, we can also model the (mutual) influences between generated emotions and actions of an agent, specifying the influence of emotion on deliberation, i.e. behaviour, action selection, planning etc.

The possibility to design discrete emotions and their specific components as explicit states or processes which are integrated into the existing (rational) models still remains unelaborated, but is a goal for future work. Although the approach presented here allows a sharp distinction, it does not enforce it, since restrictions of the flexibility might be possible. However, these restrictions should only arise from the requirements of the specific theories of emotion, which can provide the appropriate and substantial arguments.

3 Discussion

The main contribution of this paper is the E-MULAN reference architecture, which allows for the conceptual separation of concerns for emotion and cognition when modelling complex multi-agent systems. Beside the conceptual contribution our technical solution based on RENEW and CAPA allows for a direct implementation of such models.

Especially the multi-agent systems, which we usually model accordingly to the FIPA-standard with MULAN, can now be enhanced by a smooth integration of emotion models into the whole agent systems. It is important to notice that we propose to establish the possibility to apply an emotion perspective at all modelling levels of MASs: the overall system / society level, the platform / group or field level, the agent / actor level, and at the protocol level.

In this way we can investigate emotions in terms of their internal and external causes and consequences for the actor's behaviour and deliberation, for the social structure, e.g. enforcement and maintenance of social norms, or for the dynamics of the social system at the most abstract level. The implementation and analysis of the respective models, which directly use or apply the social models of emotion is still a subject to future work.

References

1. Damasio, A.R.: Looking for Spinoza: Joy, Sorrow, and the Feeling Brain. Harvest Books (2003)
2. Ekman, P., Friesen, W.V.: Unmasking the Face. Prentice Hall, Englewood Cliffs (1975)
3. Elliott, C.D.: The Affective Reasoner. A Process Model of Emotions in a Multi-Agent System. Phd thesis, Intitute for the Learning Sciences, Northwestern University (1992)
4. FIPA – homepage, http://www.fipa.org
5. Hochschild, A.R.: Emotion work, feeling rules, and social structure. American Journal of Sociology 85(3), 551–575 (1979)
6. Köhler, M., Moldt, D., Rölke, H.: A Discussion of Social Norms with Respect to the Micro-Macro Link. In: Proceedings of the 2nd International Workshop on Regulated Agent-Based Social Systems (RASTA 2003), Edinburgh (2003)
7. Köhler, M., Moldt, D., Rölke, H., Valk, R.: Linking micro and macro description of scalable social systems using reference nets. In: Fischer, K., Florian, M., Malsch, T. (eds.) Socionics. LNCS (LNAI), vol. 3413, pp. 51–67. Springer, Heidelberg (2005)
8. Kummer, O.: Referenznetze. Logos Verlag, Berlin (2002)
9. LeDoux, J.E.: Cognition and Emotion: Processing Functions and Brain Systems. In: Gazzaniga, M.S. (ed.) Handbook of Cognitive Neuroscience, pp. 357–368. Plenum Press, New York (1984)
10. Lüde, R.v., Moldt, D., Valk, R.: Sozionik: Modellierung soziologischer Theorie. Reihe: Wirtschaft – Arbeit – Technik, vol. 2. Lit-Verlag, Münster (2003)
11. Lüde, R.v., Moldt, D., Valk, R.: Selbstorganisation und Governance in knstlichen und sozialen Systemen. Reihe: Wirtschaft – Arbeit – Technik, vol. 5. Lit-Verlag, Münster (2009)
12. Renewthe reference net workshop homepage (2008), http://www.renew.de/
13. Rölke, H.: Modellierung von Agenten und Multiagentensystemen - Grundlagen und Anwendungen, vol. 2. Logos Verlag, Berlin (2004)
14. von Scheve, C., Moldt, D.: Emotion: Theoretical Investigations and Implications for Artificial Social Aggregates. In: Lindemann, G., Moldt, D., Paolucci, M. (eds.) RASTA 2002. LNCS (LNAI), vol. 2934, pp. 189–209. Springer, Heidelberg (2004)
15. Trappl, R., Petta, P., Payr, S. (eds.): Emotions in Humans and Artifacts. MIT Press, Cambridge (2003)

Towards a Taxonomy of Decision Making Problems in Multi-Agent Systems

Christian Guttmann*

School of Primary Health Care
Faculty of Medicine, Nursing and Health Sciences, Monash University
Notting Hill, 3168, VICTORIA, Australia
christian.guttmann@gmail.com

Abstract. Taxonomies in the area of Multi-Agent Systems (MAS) classify problems according to the underlying principles and assumptions of the agents' abilities, rationality and interactions. A MAS typically consists of many autonomous agents that act in highly complex, open and uncertain domains. A taxonomy can be used to make an informed choice of an efficient algorithmic solution to a class of decision making problems, but due to the complexity of the agents' reasoning and modelling abilities, building such a taxonomy is difficult. This paper addresses this complexity by placing model representation, acquisition, use and refinement at the centre of our classification. We classify problems according to four agent modelling dimensions: model of self vs. model of others, learning vs. non-learning, individual vs. group input, and competition vs. collaboration. The main contributions are extensions of existing MAS taxonomies, a description of key principles and assumptions of agent modelling, and a framework that enables a choice for an adequate approach to a given MAS decision making problem.

1 Introduction

Coordination of activities in natural and engineered systems often require that agents make individual and joint decisions. Multi-Agent Systems (MAS) consists of autonomous agents that make their own decisions (and do not follow decisions made by others) and have their own beliefs (and do not rely on beliefs maintained by other agents) [1,2]. Choosing an adequate method for effective coordination requires a thorough understanding of the underlying assumptions and principles of how agents model their social surroundings and how these models are used in decision making.

An autonomous agent requires a model of its own behaviour and that of other agents to make informed decisions about taking its next action. Knowledgeable agents (i.e., agents that have models) can predict its own actions and those of other agents. A system with such agents is structured and predictable. This is opposed to a system with ignorant agents associated with chaotic behaviour [1,2]. Research has demonstrated that using agent models benefits agent coordination in a variety of scenarios [3,4,5]. However, despite the importance of agent models in coordination, previous taxonomies do not place a notable emphasis on an agent's ability to model agent behaviour for its decision making processes. Instead, taxonomies often organise the problem space

* This research was supported in part by Linkage Grant LP0774944 from the Australian Research Council.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 195–201, 2009.

considering macro features of decision making problems. For example, [6,7] centre the issue of how many agents are required to perform one or several tasks. [8] considers how to classify different types of joint activity problems, and [9] classifies based on heterogeneity, distribution, and autonomy. This research offers useful insights into MAS coordination, but it does not emphasise a central feature of agent systems: the complexity involved when an agent maintains models of itself and others to make decisions. A better understanding of the appropriateness of an approach to a problem requires a reorganisation of the problem space based on the underlying assumptions and principles of MAS.

This paper offers extensions to existing MAS taxonomies and advances the state of the art in artificial intelligence, and particularly MAS, as follows.

- **Organisation of the space of decision making problems.** Unlike previous taxonomies, we propose a classification structure of decision making problems in MAS that places the role of agent models at its centre.
- **Analysis of the problem space.** We identify critical assumptions that define four agent modelling dimensions across the space of problems. The location of a decision making problem in this space requires an analysis against these assumptions.
- **Prescriptive framework.** Our taxonomy enables an informed choice of an approach for a given problem class as we have a clear understanding of the underlying assumptions and principles of the role of agent models in coordination.
- **Identification of research opportunities.** This taxonomy is used to classify well-known approaches, some offer provable, others heuristical solutions to problems. Our taxonomy indicates underexplored types of decision making problems.

This research arranges decision making problems in MAS emphasising the use of models maintained by agents. This taxonomy is a first attempt to find an appropriate approach for a given problem, and offers a basis to develop a unified model.

Section 2 discusses related research. Section 3 defines the four agent modelling dimensions, offers a classification scheme and positions well known existing approaches. Section 4 discusses a possible unified approach. Section 5 concludes this paper.

2 Related Research

Section 2.1 discusses the use of agent models in of MAS coordination. Section 2.2 reviews related MAS taxonomies.

2.1 Role of Agent Models

[1] argue that the coordination of MAS will be chaotic if agents are not able to predict their own behaviour and that of others. Enabling agents to maintain models of the behaviour of other agents is an important requirement for the coordination of MAS [1]. Previous research has demonstrated that using agent models benefits agent coordination in predicting the decisions made by collaborators [4], matching students with tutors in collaborative support environments [5], and predicting the performance of soccer-agents in RoboCup [3]. Each initiative makes distinct assumptions that influence what to model (feature selection), how to model it (feature representation), and how to use models (model usage). Previous research has not adequately addressed the classification of MAS decision making based on the role of agent models.

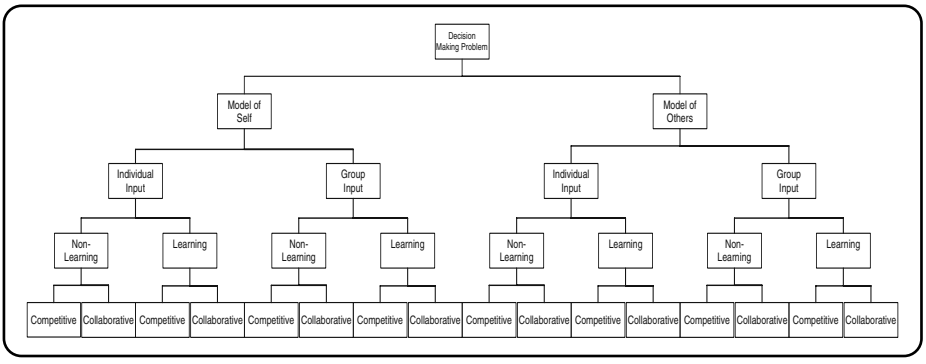


Fig. 1. A taxonomy on decision making problems in MAS

2.2 Multi-Agent System Taxonomies

Modelling the behaviour of agents is a crucial skill of an agent [1], but many MAS taxonomies do not place the role of agent models at the centre of the classification [9,6,7,8]. [7] has been widely used for task classifications and uses four categories.

	Task execution requires one agent	Task execution requires several agents
Separate task	Single Agent - Single Task (SA–ST)	Mult. Agents - Single Task (MA–ST)
Simultaneous tasks	Single Agent - Mult. Tasks (SA–MT)	Mult. Agents - Mult. Tasks (MA–MT)

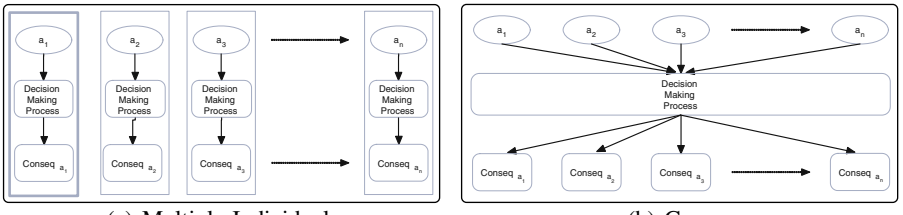
This taxonomy demonstrates how a well structured classification can assist in understanding a complex problem space. As such, the taxonomy offers a useful starting point to understand fundamental types of decision making problems. However, this MAS taxonomy (as well as many others [6,7,8,9]) has a significant limitation as the classification is not based on models maintained by agents. Our taxonomy structures this space by placing the complexity of building agent models at the taxonomy's centre.

3 Four Agent Modelling Dimensions

The complex role of models maintained by agents is central to our MAS decision making taxonomy. We consider the role of agent models in being able to make particular types of decisions. This taxonomy enables positioning of a problem in the decision making space. As with other MAS taxonomies (e.g., [6,7]), our taxonomy identifies classes of decision making problems where provable optimal solutions exist while other classes can only be solved using heuristics. While this taxonomy is not exhaustive, it characterises the complexity of agent modelling in MAS decision making problems. Four axes describe the space of MAS decision making problem (defining 16 classes, Figure 1). We now discuss a rationale for each axis.

3.1 Dimension 1: Self Model versus Model of Others

Does an agent model only its own performance or also that of others? At one extreme, an agent has information that pertains to itself, for example, it estimates its own performance or it holds a value that represents the estimated pay-off for taking a particular



(a) Multiple-Individual (b) Group

Fig. 2. Input offered by several agents (a) for each agent's decision making process with consequences for itself, or (b) for group decision making process with consequences for entire group

action. Many approaches based on each agent's knowledge of its own behaviour are *market-driven approaches*, because each agent is assumed to know its own performance best and accurately. In the Contract Net (CNET) protocol, a manager agent announces a task, each contractor assesses how well it performs the task and makes a bid, and the manager then assigns the task to the contractor which made the most adequate bid. The CNET protocol works best when the agents have accurate self-estimations, because the manager can rank the bids and select the highest bidder [10].

At the other extreme, an agent has information that pertains to other agents. Approaches where an agent requires information of other agents' behaviour are referred to as *agent modelling approaches*. Agent models are used to predict decisions of collaborators [4], match students with tutors in collaborative support environments [5], and predict the performance of soccer-agents in RoboCup [3].

3.2 Dimension 2: Individual Input versus Group Input

The input of a decision making process is either derived from an individual, or a group (Figures 2(a) and 2(b)). At one end of this spectrum, an individual agent makes a decision and uses as input for the decision making process its own knowledge (Figure 2(a)). An example here is Multi Agent Reinforcement Learning (MARL), where multiple agents execute tasks individually and use reinforcement learning to coordinate their actions, taking into account various configurations (e.g., if agents can observe each others' actions) [11,12,13]. MARL agents *do not jointly* select a team which then executes a given task, instead MARL is concerned with the internal coordination of a team.

At the other end of this spectrum, we have decision making processes that require the input of several agents (Figure 2(b)), which we refer to as *group decision making*. Voting is a preference aggregation procedure applied in situation where agents have conflicting preferences – agents "compete" as each aims to see its own preference implemented [14,15]. The aim of preference aggregation is to find a "collective decision" of what best reflects the "will" of the group.

3.3 Dimension 3: Learning versus Non-learning

We can divide this space by considering decision making problems that are made only once, or over multiple rounds where learning plays an important role [16]. In the latter case, each agent requires adequate processes to maintain models and refine them over time. For example, an agent can update its models whenever new information of its

own behaviour and that of agents is available. This information may be acquired from different sources. For example, [17]'s agents update their models using observations of other agent's behaviour over several iterations. Agent models in [5] are refined using information derived from explicit communication. Further discussions on related topics of MAS learning can be found in [16]).

Many other MAS decisions do not consider multiple rounds and learning is therefore not required. The Contract Net (CNET) protocol is an example which considers only a single round before a decision is made [10]. In particular, the CNET protocol and many of its extensions describe how a contract is made after a single announcement of the task (i.e., in a single round). In these cases, managers and contractors are not required to be able to learn. Similarly, in many voting frameworks, a group makes a decision with little consideration to long term consequences [14,15].

3.4 Dimension 4: Collaboration versus Competition

An agent's decision making style can range from being *collaborative* to being *competitive*. That is, an agent makes a decision intending to improve the welfare of a group or task (collaborative) or its own welfare (competitive).

Collaborative agents aim to maximise the welfare of the group, e.g., by finding the best allocation of a team to a task (i.e., an agent maximises the group or task utility before its own). These agents aim to offer a best "global solution" as opposed to competitive procedures that aim to find adequate trade-off's between several parties. For example, [18]'s agents are collaborative, because each agent aims to detect and resolve problems that could jeopardise a successful completion of a mission.

A competitive agent exhibits behaviour that maximises its own utility – a behaviour also referred to as *self-rational or self-interested*. Agents exhibit self-rational behaviour in settings where resources are limited or different agents have opposing or conflicting beliefs. For example, in auctioning, a group decision is made based on the competitive bids of agents [19,20]. In these settings, an agent may only have information of its own evaluation of the auctioned item in question, but information by competing agents may be unreliable and its accuracy can not be trusted.

4 A Unified Approach to Distributed Decision Making?

Can we find a unified approach that represents and solves the problem classes defined in our taxonomy? Our taxonomy shows that there is a multitude of MAS approaches for decision making, and many are located at different ends of the dimensions as we discussed in Section 3. How can we build a computational model that unifies many, if not all approaches to decision making problems classified in our taxonomy?

[21] offers an initial framework that captures a wide variety of decision making problems located across the dimensions discussed in Section 3. [21] studies the refinements of allocations based on group decisions. We refer to this as Collective Iterative Allocation (CIA), because decisions are made together and allocations can be refined (and iterated over time). In CIA, agents model their own performance (as in CNET) and that of others. It allows for single round decision problems as well as for multiple rounds (where agents are able to learn). Different competition and collaboration approaches can be defined by the group decision policy. The CIA framework assumes that a decision is always made by a group (that is, the voting policy requires input from several

agents). One way to address this issue is to consider the conditions under which each agent should make its own decisions (e.g., as is done in MARL) or follow the decisions made by the group decision policy.

5 Conclusion

This paper discusses a taxonomy for MAS decision making problems. This paper offers extensions to existing taxonomies on decision making in MAS and makes four contributions. We propose a classification structure of decision making problems in MAS that places the role of agent models at its centre (this classification can be represented using a tree structure where classes are clearly separated). We identified critical assumptions which define four agent modelling dimensions in the space of problems. A location of a decision making problem in this space requires an analysis against these assumptions. Our taxonomy enables an informed choice of an approach for a given problem class as we have a clearer understanding of the underlying assumptions and principles of the role of agent models in coordination. Finally, this taxonomy can be used to identify research opportunities as it classifies well-known approaches. We also discussed that the CIA framework is a first step towards a unified approach for many decision making problems. A future research direction is to extend this framework to enable further unification. In future works, we aim to offer a comprehensive survey of existing approaches, as well as to continue the formalisation of the taxonomy discussed in this paper.

References

1. Bond, A.H., Gasser, L.: An analysis of problems and research in DAI. In: Bond, A.H., Gasser, L. (eds.) Readings in Distributed Artificial Intelligence (1988)
2. Wooldridge, M.: Introduction to Multiagent Systems. John Wiley & Sons, Inc., Chichester (2002)
3. Stone, P., Riley, P., Veloso, M.M.: Defining and using ideal teammate and opponent agent models. In: Proceedings of the Innovative Applications of Artificial Intelligence Conference (IAAI), pp. 1040–1045 (2000)
4. Gmytrasiewicz, P.J., Durfee, E.H.: Rational communication in multi-agent environments. Autonomous Agents and Multi-Agent Systems 4(3), 233–272 (2001)
5. Vassileva, J., McCalla, G.I., Greer, J.E.: Multi-agent multi-user modeling in I-Help. User Modeling and User-Adapted Interaction 13(1–2), 179–210 (2003)
6. Dudek, G., Jenkin, M., Milios, E., Wilkes, D.: A taxonomy for multi-agent robotics. Autonomous Robots 3(4), 375–397 (1996)
7. Gerkey, B., Mataric, M.: Are (explicit) multi-robot coordination and multi-agent coordination really so different. In: Proceedings of the AAAI Spring Symposium on Bridging the Multi-agent and Multi-robotic Research Gap, pp. 1–3 (2004)
8. Klein, G., Feltovich, P., Bradshaw, J., Woods, D.: Common ground and coordination in joint activity. Organizational Simulation (2004)
9. Bird, S.: Toward a taxonomy of multi-agent systems. International Journal of Man-Machine Studies 39(4), 689–704 (1993)
10. Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers 29(12), 1104–1113 (1980)
11. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI), pp. 746–752 (1998)

12. Shoham, Y., Powers, R., Grenager, T.: Multi-agent reinforcement learning: A critical survey. In: AAAI Fall Symposium on Artificial Multi-Agent Learning (2004)
13. Sandholm, T.: Perspectives on Multiagent Learning. Artificial Intelligence (Special Issue on Multiagent Learning) 171, 382–391 (2007)
14. Arrow, K.J.: Social choice and individual values. J. Wiley, New York (1951)
15. Fishburn, P.: The theory of social choice. Princeton University Press, Princeton (1973)
16. Stone, P., Veloso, M.M.: Multiagent systems: A survey from a machine learning perspective. Autonomous Robots 8(3), 345–383 (2000)
17. Suryadi, D., Gmytrasiewicz, P.J.: Learning models of other agents using influence diagrams. In: Proceedings of the seventh International Conference on User Modeling (UM), Banff, Canada, pp. 223–232 (1999)
18. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
19. Vickrey, W.: Counterspeculation, Auctions, and Competitive Sealed Tenders. The Journal of Finance 16(1), 8–37 (1961)
20. Boutilier, C., Goldszmidt, M., Sabata, B.: Sequential auctions for the allocation of resources with complementarities. In: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI), pp. 527–523 (1999)
21. Guttmann, C.: Collective Iterative Allocation. PhD thesis, Monash University (2008)

Modeling Tools for Platform Specific Design of Multi-Agent Systems

Geylani Kardas[1], Erdem Eser Ekinci[2], Bekir Afsar[2], Oguz Dikenelli[2], and N. Yasemin Topaloglu[2]

[1] Ege University, International Computer Institute, 35100 Bornova, Izmir, Turkey
geylani.kardas@ege.edu.tr
[2] Ege University, Department of Computer Engineering, 35100 Bornova, Izmir, Turkey
erdemeserekinci@gmail.com,
{bekir.afsar,oguz.dikenelli,yasemin.topaloglu}@ege.edu.tr

Abstract. In this paper, we introduce platform specific modeling and code generation tools for the model driven development of multi-agent systems (MAS). These tools enable agent developers to model their MASs for the SEAGENT and the JADEX agent platforms based on the semantics and design principles of these platforms. The toolkit also provides automatic code generation for agent developers in order to implement their MASs on the target platforms. Generated codes may vary on type (e.g. Java class files, XML documents or ontologies) according to each platform's requirements.

1 Introduction

Model Driven Development (MDD), which aims to change the focus of software development from code to models, may also provide rapid and easy development of Multi-agent Systems (MAS). However, such a development process should be supported by modeling tools in order to assist developers during their design. Many researchers in Agent-oriented software engineering (AOSE) community propose model driven approaches (e.g. [1], [2] and [3]) in MAS development and also introduce related modeling tools for their approaches. This study contributes to these efforts by introducing new graphical modeling tools for different MAS platforms.

Based on the well-known MDD realization called Model Driven Architecture (MDA)[1], our ongoing work aims to define a MAS development process which will consider the ontologies as the basic components of the MAS architecture. The proposed development process includes definition of metamodels for each layer of the MDA architecture; called *the Computation Independent Model (CIM)*, *the Platform Independent Model (PIM)*, and *the Platform Specific Model (PSM)* and provides modeling software tools for modeling in each layer. This study introduces the software modeling tools which can be used at the PSM level. The developers can use these tools to model MASs for SEAGENT [4] and JADEX [5] platforms and obtain auto-generated software codes of their agent systems for the related platforms.

[1] http://www.omg.org/mda/

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 202–207, 2009.

The paper is organized as follows: In Section 2, we briefly discuss design and use of the tools. Visual modeling and code generation for SEAGENT and JADEX agents are discussed in Section 3 and 4 respectively. Section 5 covers related work. Conclusion and future work are given in Section 6.

2 Design and Use of the Modeling Tools

The modeling tools introduced in this paper are developed on Eclipse[2] platform by using Graphical Modeling Framework (GMF)[3]. GMF is a framework for building graphical modeling editors for various domains. In our study, we (1) provide domain models of SEAGENT and JADEX agent platforms as Ecore[4] metamodels, (2) prepare graphical elements representing the agent domain elements and their relations, (3) map agent components with the related graphical nodes and (4) generate the agent modeling editors as the Eclipse plug-ins.

The developers use editors for visually modeling their agent systems. Agent domain elements and their relationship links are represented in the editor palettes. The developers choose desired elements and links from palettes and visually create their agent models as will be discussed in the following sections. The editor environment also supports developers in model consistency and prevents wrong relation establishments between agent model elements.

The outputs of the visual modeling are the model documents for the designed agent systems. The next step is the automatic generation of agent software codes, ontology documents and any other system files from visually created agent models. We employ the Abstract Syntax Tree (AST) and related parser in the Eclipse Java Development Tools (JDT) for automatic generation of SEAGENT agent software, plan documents and ontology files. On the other hand, MOFScript[5] is used to generate JADEX agent description files and agent plan codes. Above mentioned code generations are completely abstract from the developers and hence developers do not deal with the generation process.

In order to illustrate practical use of the introduced tools, let us consider a multi-agent based e-barter system. A barter system is an alternative commerce approach where customers meet at a marketplace in order to exchange their goods or services without currency. An agent-based e-barter system consists of agents called *Customer* that exchange goods or services of owners corresponding to their preferences. The *Barter Manager* agent manages all trades in the system. This agent is responsible for collecting barter proposals, matching proper barter proposals and tracking the bargaining process between customer agents. In the following sections, the registration scenario of customer agents with the Barter Manager agent is discussed for the demonstration of the modeling tools. Interested readers may refer to [6] for the complete design and description of the related e-barter MAS system.

[2] http://www.eclipse.org/
[3] http://www.eclipse.org/gmf/
[4] http://www.eclipse.org/modeling/emf/
[5] http://www.eclipse.org/gmt/mofscript/

3 Modeling SEAGENT Agents

SEAGENT [4] is an agent development platform in which Semantic Web enabled MASs can be developed in an interactive and test-driven manner. SEAGENT Agents manage all of their internal knowledge using OWL[6] ontologies and can interact with the semantic web services. Development of SEAGENT MASs includes MAS modeling according to 4 viewpoints called *Organization*, *Plan*, *Protocol* and *Domain*. Within *Organization Model*, agents, organizations, roles, goals and responsibilities of the roles are declared. *Plan Model* provides internal planning of the agents based on the Hierarchical Task Network (HTN) [7] paradigm. *Protocol Models* include protocols for the interactions between SEAGENT agents. Finally, *Domain Model* composes the ontological representation of the related business domain elements**.** Our modeling toolkit includes editors for all 4 base models of the SEAGENT. However, due to space limitations only editor for the Plan model is discussed in here.

In the SEAGENT framework, agents execute their tasks according to HTN. HTN planning creates plans by task decomposition. This decomposition process continues until the planning system finds primitive tasks that can be performed directly. In the HTN formalism, there are two types of tasks: complex tasks called *behaviours* and primitive tasks called *actions*. Each plan has a root task which is a behaviour itself consisting of subtasks (actions) that are composed to achieve a predefined goal. Behaviours hold a "reduction schema" data structure that defines the decomposition of the complex task to subtasks and the information flow between these subtasks and their parent task. Actions, on the other hand, are primitive tasks that can be executed by the SEAGENT planner using the Java Reflection API.

Our toolkit provides the editor called HTN planner for the visual modeling of the SEAGENT agent plans (Fig. 1). In this editor, agent developers can create plan models including agent behaviours, actions and linkage between behaviour and their subtasks by first selecting proper elements from the component palette and then entering their attribute values. Fig. 1 also depicts the HTN plan model of the Barter Manager agent for the customer registration scenario. The Barter Manager has the behaviour "BHResponseRegistration" which can be decomposed into three actions called "ACCheckExistence", "ACRegisterCompany" and "ACAcceptCompany Registration". In these actions, the agent first controls whether the customer already has been registered, then checks customer's registration data according to the related ontology if it is a new customer and finally realizes the registration.

SEAGENT platform executes the artifacts which are stored in two different representation formats: OWL and Java. Considering HTN plans of SEAGENT agents, our editor has built-in model to text translators for automatic generalization of agent behaviour ontologies and template Java class codes of the related behaviour's actions. As shown in Fig. 1, the developer just needs to right-click on the proper model element, chose code generation and set required parameters (e.g. source code folder, namespace and action method name). Then the editor automatically generates required plan ontologies (for the root behaviours) and action Java codes (for subtasks).

[6] http://www.w3.org/2004/OWL/

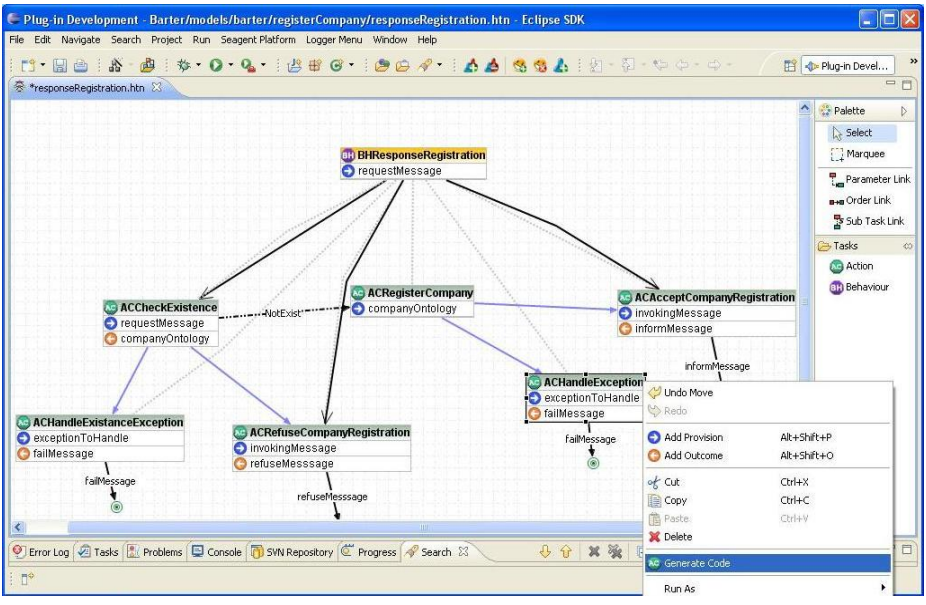


Fig. 1. Modeling SEAGENT Agent HTN plans

4 Modeling JADEX Agents

Our toolkit includes modeling and automatic code generation utilities for another agent development platform called JADEX. JADEX [5] provides an engine and a programming platform for developing well-known Belief-Desire-Intention (BDI) [8] agents. The development of JADEX agents is based on a hybrid approach in which declaration of static agent properties and programming of executable agent plans take place. Declaration of static agent properties is given in files called Agent Definition Files (ADF). An ADF file is written using XML and specifies the BDI model of the related agent. On the other hand, agent plans are executable components and they are given in Java program files. In order to assist agent developers in design and development of JADEX BDI agents, we provided a graphical modeling editor and an automatic code generation tool based on the design principles discussed in Section 2. Developers can model the BDI architecture of the agents by choosing appropriate model elements from the component palette, drawing the relationship links between elements and setting model attributes. The editor environment (Fig. 2) provides visualization of the BDI model of the agents and supports easy and efficient development of JADEX agents. The editor also prevents users from wrong relationship establishments between BDI elements. For example, *Body* of the plan node can only be linked with the *Plan* node according to the BDI semantics of the JADEX metamodel. Hence the editor does not allow linking *Body* nodes with any other nodes except the Plan node.

Considering our case study, the BDI model of the Barter Manager agent can be visualized as given in Fig. 2. Registration information received from the customer agents is given as the initial facts (beliefs) for the Barter Manager (at the upper left of

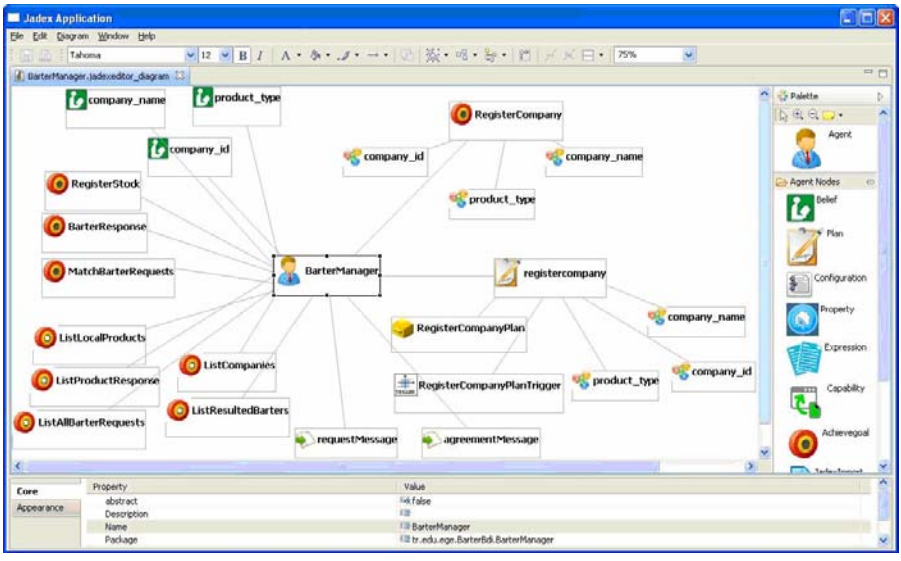


Fig. 2. Modeling JADEX BDI Agents

the model in Fig. 2). The model also includes agent's goals to be achieved (at the lower left of the model), the plan structure (at the right of the model) and their relations. After modeling of the JADEX Agent system is completed, developers can obtain the ADF and related agent plan's Java classes by using the integrated code generator. Agent BDI models are stored as Ecore files. The code generator uses these files as input, applies a model-to-text transformation using MOFScript and finally outputs related ADF and template Java class files for the designed agent system.

5 Related Work

Several MAS software tools exist for modeling agent systems according to various AOSE methodologies e.g. agentTool[7] for O-Mase, IDK[8] for INGENIAS, TAOM4e[9] for Tropos and PDT[10] for Prometheus. These tools mostly cover analysis and design of MASs and a few of them only consider implementation and provide code generation for specific agent platforms. On the other hand, studies like [1] and [2] utilize tools for MDD of MASs but only one MAS development platform and its PSM are taken into consideration. The graphical modeling editor introduced in [9] has the same design principles with our toolkit. The editor is GMF based and supports MDD of MASs at the PIM level. The toolkit introduced in this paper contributes to those noteworthy studies by supporting platform specific modeling and implementation of MASs according to MDA principles.

[7] http://agenttool.cis.ksu.edu/
[8] http://sourceforge.net/projects/ingenias/
[9] http://sra.itc.it/tools/taom4e/
[10] http://www.cs.rmit.edu.au/agents/pdt/

6 Conclusion and Future Work

In this paper, platform specific modeling and code generation tools for the MDD of SEAGENT and JADEX agents are introduced. These tools enable agent developers to design and implement their MASs on different target platforms which vary on the design principles such as BDI, HTN and Semantic Web integration. In our future work, we plan to develop another modeling toolkit for the platform independent modeling of MASs. The toolkit will be again based on the GMF and use the enhanced version of the PIM discussed in [3]. This new toolkit will be integrated into the development environment introduced in this paper.

Acknowledgements

This work is funded by The Scientific and Technological Research Council of Turkey (TUBITAK) Electric, Electronic and Informatics Research Group (EEEAG) under grant 108E141.

References

[1] Perini, A., Susi, A.: Automating Model Transformations in Agent-Oriented Modeling. In: Müller, J.P., Zambonelli, F. (eds.) AOSE 2005. LNCS, vol. 3950, pp. 167–178. Springer, Heidelberg (2006)
[2] Pavon, J., Gomez, J., Fuentes, R.: Model Driven Development of Multi-Agent Systems. In: Rensink, A., Warmer, J. (eds.) ECMDA-FA 2006. LNCS, vol. 4066, pp. 284–298. Springer, Heidelberg (2006)
[3] Kardas, G., Goknil, A., Dikenelli, O., Topaloglu, N.Y.: Modeling the Interaction between Semantic Agents and Semantic Web Services using MDA Approach. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 209–228. Springer, Heidelberg (2007)
[4] SEAGENT MAS Development Framework, http://seagent.ege.edu.tr/
[5] JADEX BDI Agent System, http://jadex.informatik.uni-hamburg.de/
[6] Cakirlar, I., Ekinci, E.E., Dikenelli, O.: Exception Handling in Multi-Agent Systems. In: 9th Workshop on Engineering Societies in the Agents World, Saint-Etienne (2008)
[7] Williamson, M., Decker, K., Sycara, K.: Unified Information and Control Flow in Hierarchical Task Networks. In: AAAI 1996 Workshop, pp. 142–150 (1996)
[8] Rao, A., Georgeff, M.: BDI Agents: From Theory to Practice. In: First International Conference on Multi-Agent Systems, San Francisco, pp. 312–319 (1995)
[9] Warwas, S., Hahn, C.: The concrete syntax of the platform independent modeling language for multiagent systems. In: ATOP 2008 Workshop, Estoril, pp. 94–105 (2008)

L2-SVM Training with Distributed Data

Stefano Lodi[1], Ricardo Ñanculef[2], and Claudio Sartori[1]

[1] Dept. of Electronics, Comp. Sc. and Systems, University of Bologna, Italy
[2] Department of Informatics, Federico Santa María University, Chile

Abstract. We propose an algorithm for the problem of training a SVM model when the set of training examples is horizontally distributed across several data sources. The algorithm requires only one pass through each remote source of training examples, and its accuracy and efficiency follow a clear pattern as function of a user-defined parameter. We outline an agent-based implementation of the algorithm.

1 Introduction

The Support Vector Machine (SVM) [1] is one of the most effective methods to learn classifiers from data, and is currently used in a number of real-world applications. In this work, we address the problem of distributed SVM learning, i.e., learning a SVM when the examples are fragmented among the sites of a network, and the amount of data which is transmitted over the network must be as small as possible, to meet constraints of autonomy, scalability, and transmission load. Under such constraints, transmitting the data to a single site and applying a centralized method on the collected data is clearly an infeasible approach. On the other hand, distributed algorithms for the problem currently require several accesses to the data sources in order to achieve a reasonable approximation to the centralized solution [2,3]. The algorithm we propose requires in contrast only one access to each remote location and can be as accurate as desired, trading off complexity and accuracy.

2 SVMs and Minimal Enclosing Balls

Support Vector Machines (SVMs) [4] address the problem of binary classification by building a hyperplane to represent the boundary between the two classes. This hyperplane ($\mathbf{w}^T\mathbf{z} + b = 0$) is built in a feature space $Z = \phi(\mathcal{X})$ implicitly induced from $\mathcal{X}$ by means of a kernel function k which computes the dot products $\mathbf{z}_1^T\mathbf{z}_2 = \phi(\mathbf{x}_1)^T\phi(\mathbf{x}_2)$ in Z directly on $\mathcal{X}$. The so called L2-SVM builds the separating hyperplane by solving the following convex optimization problem:

$$\begin{gathered}\mathbf{min}(\alpha) : \sum_{i,j\in I} \alpha_i\alpha_j \left(y_i y_j k(\mathbf{x}_i,\mathbf{x}_j) + y_i y_j + \frac{\delta_{ij}}{C}\right) \\ \mathbf{st:}\ 0 \le \alpha_i,\ \sum_i \alpha_i = 1\end{gathered} \tag{1}$$

The parameter C trades off training accuracy and smoothness of the solution. Its optimal value is determined using model selection techniques and depends on

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 208–213, 2009.

the degree of noise and overlap among the classes [4]. From the solution α the hyperplane parameters as recovered as $w = \sum_i y_i \alpha_i \mathbf{z}_i$ and $b = \sum_i \alpha_i y_i$. Note that the solution finally depends only on the examples for which $\alpha_i \neq 0$ which are called the *support vectors*.

Although the L2-SVM is slightly different from the original SVM formulation, both models obtain comparable performance in practice [5]. As shown in [5] the main appeal of the L2 implementation is that it supports a convenient reduction to a minimal enclosing ball (MEB) problem when the kernel used in the SVM is normalized, that is $k(\mathbf{x}, \mathbf{x}) = \Delta \; \forall x \in \mathcal{X}$. The advantage of this equivalence is that the Bădoiu and Clarkson algorithm [6] can efficiently approximate the solution of a MEB problem with any degree of accuracy.

The ball $\mathcal{B}(\mathbf{c}, R)$ of center $\mathbf{c} \in \tilde{Z}$ and radius R in $\mathbb{R}$ is the subset of points $\tilde{z} \in \tilde{Z}$ for which $\|\tilde{\mathbf{z}} - \mathbf{c}\|^2 \leq R^2$. The *minimal enclosing ball* of a set of points $S = \{\tilde{\mathbf{z}}_i : i \in I\}$ in $\tilde{Z}$ is in turn the ball $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$ of smallest radius that contains S, that is the solution to the following optimization problem:

$$\begin{aligned} &\mathbf{min}(\mathbf{R}, \mathbf{c}) : R^2 \\ &\mathbf{st:}\ \|\tilde{\mathbf{z}} - \mathbf{c}\|^2 \leq R^2 \;\; \forall \tilde{\mathbf{z}} \in S \end{aligned} \tag{2}$$

The algorithm of Bădoiu and Clarkson [6] to approximate the solution to this problem exploits the ideas of core-set and ϵ-approximation to the minimal enclosing ball of a set of points. A set $\mathcal{C}_S \subset S$ will be called a *core-set* of S if the minimal enclosing ball computed over $\mathcal{C}_S$ is equivalent to the minimal enclosing ball considering all the points in S. A ball $\mathcal{B}(\mathbf{c}, R)$ is said a ϵ-*approximation* to the minimal enclosing ball $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$ of S if $R \leq R^*$ and it contains S up to precision ϵ, that is $S \subset \mathcal{B}(\mathbf{c}, (1+\epsilon)R)$. Consequently, a set $\mathcal{C}_{S,\epsilon}$ is called a ϵ-*core-set* if the minimal enclosing ball of $\mathcal{C}_{S,\epsilon}$ is a ϵ-approximation to $\mathcal{B}^*(S, \mathbf{c}^*, R^*)$. Now, the algorithm of Bădoiu and Clarkson is a greedy approach to find a ϵ-core-set of S, which converges in no more than $O(\frac{1}{\epsilon})$ iterations. Since each iteration adds only one point to the core-set, the final size of the core-set is also $O(\frac{1}{\epsilon})$. Hence, the accuracy/complexity tradeoff of the obtained solution monotonically depends on ϵ.

3 Distributed Learning of the L2-SVM

We assume a collection of datasets $S_j = \{(\mathbf{x}_i^j, y_i^j); i \in I_j\}$, $j = 1, \ldots, p$, and a pool of *nodes* $\mathcal{N} = \{N_1, \ldots, N_p\}$ such that S_j is located in N_j. Our task is to train a SVM model on the union of the datasets $S = \bigcup_{j=1}^{p} S_j$. To this purpose we identify a *coordinator node* C which serves as an organizer of local learning tasks and integrator of the results of the remote nodes.

A first approach to the task could be the following: since the SVM only depends on the support vectors, we can compute local SVMs at each node, identify the local support vectors SV_j and send them to the coordinator node, which finally builds a SVM with the union of the support sets. This algorithm requires only one access to the remote nodes. However, it can be shown for the L1-SVM that the union of the SV_j does not coincide with the support vectors identified when training the SVM with the complete training set [2,7].

Our method is based in contrast on the following observation.

Proposition 1. *Let $\tilde{Z}$ be a normalized dot-product space, $S = S_1 \cup \ldots \cup S_p \subset \tilde{Z}$ and $\mathcal{C}_j$ a core-set for S_j, $j = 1, \ldots, p$,. Then $\mathcal{C}_S = \bigcup_{j=1}^{p} \mathcal{C}_j$ is a core-set for S.*

Recall from Section 2 that for a kernel-induced normalized feature space, training a L2-SVM on a dataset D is equivalent to building a minimal enclosing ball of D. Then, if we are able to first compute a core-set for S we can then build the ball only with the points in the core-set. However, if D is the union of the remote subsets by Proposition 1 we can compute core-sets for each remote D_j and then join the core-sets to build the SVM. If the computation of the remote core-sets is exact the final computation of the SVM is also correct and we only need one access to the remote nodes. Moreover, since the local computation of the core-sets is only dependent on the local training set, neither communication nor data exchange is required among the nodes. This contrasts with other methods which iterate among the remote nodes until convergence [2], mimicking working-set selection methods for (centralized) training of SVMs [4].

Algorithm (1) summarizes our procedure. To turn this exact procedure into an efficient one in terms of both the amount of computation carried out by local nodes and the amount of data sent to the coordinator, instead of computing an exact core-set at each remote node we proceed using the Bădoiu-Clarkson algorithm. This method approximates the core-set with any degree of accuracy ϵ. Both the time incurred in the computation and the final size of the core-sets grow as $O(\frac{1}{\epsilon})$. Thus, ϵ directly controls the tradeoff between efficiency and exactness of our method.

Algorithm 1. Distributed Algorithm for Training the L2-SVM

1: Estimate the hyper-parameters C and kernel-specific parameters.
2: **for** Each Remote Node $i = 1, \ldots, p$
3: Receive hyper-parameters from the coordinator. If not available estimate them locally.
4: Compute a core-set $\mathcal{C}_i$ of S_i using the corresponding kernel and parameters.
5: Send the core-set to the coordinator.
6: **end for**
7: Join the core-sets $\mathcal{C}_S = \mathcal{C}_1 \cup \ldots \cup \mathcal{C}_p$
8: Compute the minimal-enclosing-ball of C_S.

4 Experiments

We provide simulations of our procedure in two classification problems: *Breast-Cancer-Wisconsin* and *Handwritten-Digits* available and described in [8]. Since the first problem is multi-class we select the digits 1 and 7, as in previous research, to obtain a binary problem. Both datasets are partitioned randomly among a set of 5 nodes. Then we analyze the scalability of the method in the number of nodes.

The statistics to consider at each experiment will be computed after 50 trials. At each repetition an independent random sample corresponding to the 85% of the available examples will be selected as training set. All the SVMs are trained using a gaussian kernel $k(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\|\mathbf{x}_1 - \mathbf{x}_2\|^2/\sigma^2)$ and the scale parameter σ^2 is estimated as the averaged distance among training examples.

We contrast the distributed solution with the full centralized method which imports all the data from the remote nodes to train a L2-SVM model. Additionally, we include the centralized L1-SVM in the analysis. Our research hypothesis is that the distributed solution can closely approximate the full centralized method even with a lower network load.

Figure (1) compares the average testing among the trials against the value of hyper-parameter C which is usually determined using a model selection approach. Means are depicted with two standard deviations around. We can observe that the greater the precision with which the remote nodes solve the local learning tasks, the greater the precision with which the distributed solution approximates the centralized solution. For the greatest precision considered in this work (10^{-5}) the differences between the means differs in around 0.1% of correct classification in the handwritten-digits experiment and 2% or less in the breast-cancer problem. The differences tend to decrease if we focus in the values of C which give the best classification results, which are the targets of the model selection approach used to determine this parameter. Finally, except for lower values of ϵ, the variances of centralized and distributed solutions (among trials) also follow a common pattern. Figure (2) shows the fraction of the data imported from remote nodes by the distributed solution. For a reasonable precision of 10^{-3} we obtain a reduction of 80% in the digit recognition problem and

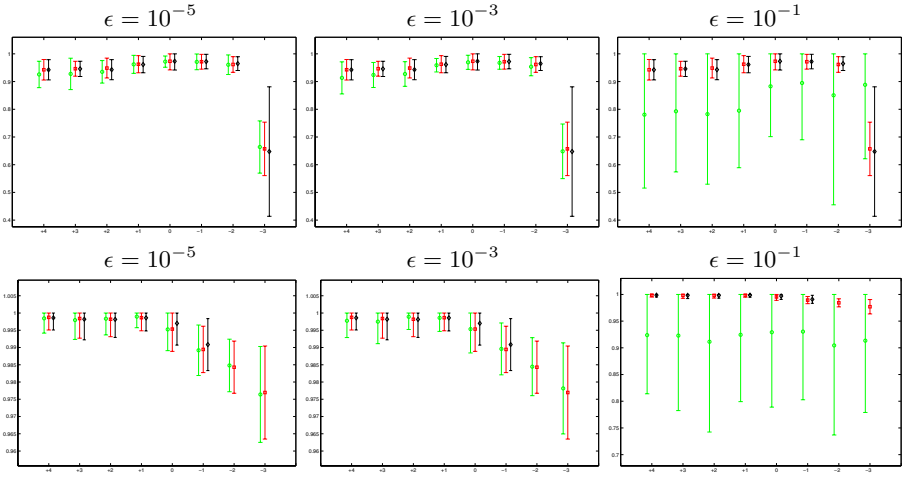


Fig. 1. Testing Accuracy (y axis) vs. value of **Parameter C** (x axis, $C = 10^4 : 10^{-3}$) for different values of ϵ. First row: *cancer* experiment. Second row: *handwritten digits* experiment. Circles: distributed L2-SVM; Squares: centralized L2-SVM; Diamonds: centralized L1-SVM.

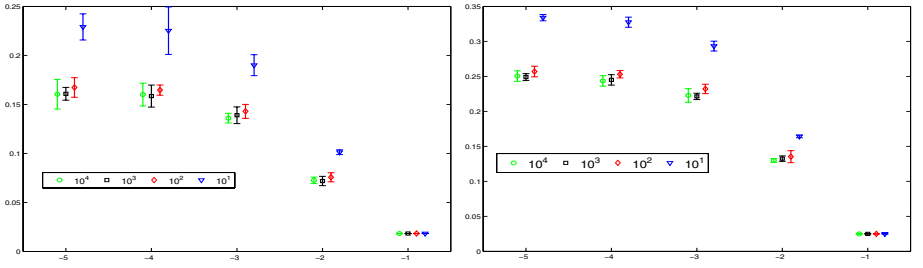


Fig. 2. Fraction of data imported from remote nodes (y axis) vs. **Precision** ϵ of the enclosing-balls (x axis, $\epsilon = 10^{-5} : 10^{-1}$) for different values of $C = 10^5 : 10^1$. First column: *cancer* problem. Second column: *handwritten digits* problem.

85% in the breast-cancer problem comparing with the full centralized solutions which need to import all the data. Figure (3) additionally shows the dependence of training time and precision ϵ of the remote computations. We conclude that ϵ monotonically controls the tradeoff accuracy/efficiency.

Finally, we carried out a short experiment to measure the scalability of the method if we change the number of nodes with the handwritten-digits dataset and a fixed value of C ($= 10$). We can observe that accuracy is highly independent of the number of nodes, whereas the total amount of data imported from them tends to decrease with the number of nodes.

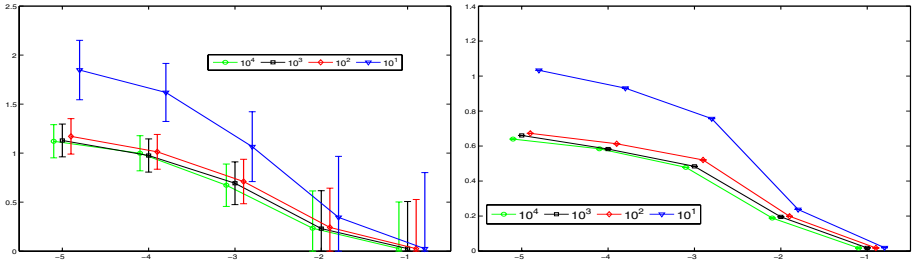


Fig. 3. Training Time (y axis) **versus Precision** ϵ (x axis, $\epsilon = 10^{-5} : 10^{-1}$) for different values of $C = 10^4 : 10^1$. First column: *cancer* problem. Second column: *handwritten digits* problem. The figure overlaps.

5 Agent-Oriented Negotiation of Parameters

Although the sites where the distributed data reside are co-interested in the construction of an accurate overall model, they might also be in conflict for setting the global parameter C, which influences the performance of the SVM model in a data-dependent way. Different sites may attain optimal model performance for different valutes of C, and the optimal value for the overall model may be still different. Also the parameter ϵ controls the accuracy of the model; however, it affects the final solution monotonically and all the sites in a similar way. Therefore, negotiation is not crucial for ϵ. For C, negotiation is needed to guarantee

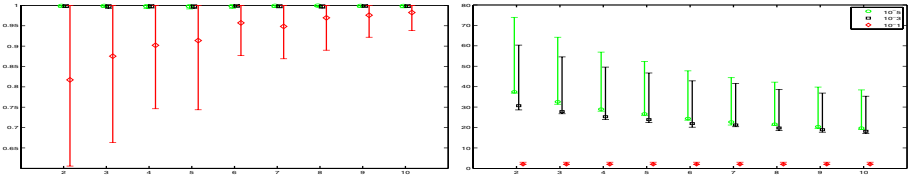


Fig. 4. Accuracy (y axis, left) and **Total N of Examples Imported** (y axis,right) **versus N of Nodes** (2 to 10) for the *digits* problem. Results for different precisions ϵ: 10^{-5} (circles), 10^{-3} (squares) and 10^{-1} (diamonds).

that a single value falling inside the set of admissibile values specified by each site is chosen. To this end, we assume exactly one agent for each site, including the coordinator, manages all interaction with the other sites and the needed iteration through different values of C.

Initially, the coordinator sends a call for proposal to every site agent for an admissibile set of intervals for C, with a deadline for replying. The agents compute locally SVM models for an autonomously chosen set of values of C and send a proposal consisting of the two extremal values of every contiguous set of C values satisfying a threshold on model accuracy. The coordinator intersects the intervals; on non-empty intersection, sends a proposal with the mean of the intersection; otherwise, informs the pairs of agents proposing disjoint intervals a negotiation is required to extend the nearest intervals bounds until an overlap is obtained. At the end of each negotiation, the parties inform the coordinator of the resulting interval. The coordinator finally proposes the final value for C and upon accept from all agents, informs all agents. The agents compute the core-sets and inform of their content the coordinator, which computes the global model using the final value for C.

References

1. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. Annals of Statistics 36(3), 1171–1220 (2008)
2. Caragea, D.: Learning Classifiers from Distributed, Semantically Heterogeneous, Autonomous Data Sources. PhD thesis, Iowa State University (2004)
3. Hazan, T., Man, A., Shashua, A.: A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality, pp. 1–8 (June 2008)
4. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
5. Tsang, I., Kwok, J., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. Journal of Machine Learning Research 6, 363–392 (2005)
6. Bădoiu, M., Clarkson, K.L.: Optimal core-sets for balls. Comput. Geom. Theory Appl. 40(1), 14–22 (2008)
7. Syed, N.A., Liu, H., Sung, K.K.: Incremental learning with support vector machines. In: Proc. of the 16^{th} IJCAI (1999)
8. Asuncion, A., Newman, D.: UCI machine learning repository (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html

Framework for Dynamic Life Critical Situations Using Agents

Jenny Lundberg[1] and Anne Håkansson[2]

[1] Blekinge Institute of Technology
Box 520, 37225 Ronneby, Sweden
[2] Computer and Systems Sciences
Forum 100, SE-164 40 Kista, Sweden
jlu@bth.se, anneh@dsv.su.se

Abstract. In this paper, we present a framework incorporating a multi-agent system (MAS) that enables aid for international effect-based operation in emergency situations. The outcome is to empower emergency personnel, which can support collaboration between different international services by informing them about the emergency, matching competences and resources of the teams and volunteers. The challenge in emergency contexts is the abbreviations forming an information-carrying structure, which is especially important when abbreviations are exchanged between different services like rescue, military and emergency. We propose a framework, which provides the right information, rescue team, and services at the right place. The MAS can support information dissemination in dynamic situations in context, based on the information extraction and matching of the contents of the underlying ontologies. In the framework the system poses a sensible solution to the international rescue teams' need of a high quality handling of life critical situations.

Keywords: Multi-agent systems, intelligent agents, emergency.

1 Introduction

International support in emergency services is necessary to handle natural catastrophes like the earthquake creating a Tsunami in the Indian Ocean 2004. Many countries were suffering from this catastrophe, with approximately 223.000 fatalities, mainly due to citizens were taking vacation on the coasts of South-East Asia. The rescue actions from the Swedish government failed and from an investigation it was found that it suffered criticism due to the lack of an organisation managing such serious crisis. But the criticism also concerned lack of understanding the extent of the disaster, the collection of relevant information and coordination of measures [1]. Presumably, a closer connection between international rescue organisations in the involved countries can build a ground for a solid unified action of accurate force to meet disasters that pose immediate negative effects on several countries. Therefore, we need proper ICT systems to support disaster relief [10]. Multi-agent systems in emergency contexts have been used [2] [11] [12], but it is desirable to develop a framework with agents and ontologies to handle dynamic emergency contexts.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 214–219, 2009.

To receive comprehensive information and disseminate it fast, the emergency operators write abbreviations in the system, which is translated into for nationally rescue teams, commonly utilised and understandable language. This language needs to be interpreted by international rescue teams. We suggest using Multi-agent system (MAS) to support the dissemination of the information to international teams. From the input of comprehensive information to MAS, the agents handle the conversion of abbreviations into understandable language and translations into other languages.

In dynamic emergency areas, the use of acronym and abbreviations are widely spread, such as, Air Traffic Control, operator control of critical infrastructures and several other medical applications [9]. Abbreviations are context sensitive [13], and as early as 1975, Woods [3] stated the importance of semantics and not only syntax. In this paper, we work with an awareness of syntax and semantics as we apply multi-agent systems in the real world context of international emergency handling.

2 Scenario

To develop MAS, we look at a future scenario based on the real world Tsunami 2004: Sweden and other related countries become alerted about the Tsunami through the early warning systems, and establish contacts with Thailand's SOS centres, simultaneously as the earthquake starts, which is approximately 2 hours before the wave hit the shore. Sweden sends Jumbo jets with rescue service and nursing staff, and communicates with Thailand via international language on board the Jumbo jet. They plan the rescue action and make strategies for the rescue, match competences and communicate with local rescue services. When the Swedish rescue force arrives, vehicles are waiting for them for quick transport to rescue areas, hospitals and collapsed buildings. Swedish SOS (at the scene and in Sweden) must control which other international resources are available and match their recourses and competences**.** They must also identify a large number of volunteers without rescue education. These will be given quick training on place. Since it is a tsunami this training is lead by rescue competent personnel. The matching of rescue competence against the requirements in the action plan is made in a unified process including search for proper rescue leading competences. Furthermore, when all the rescue teams arrive on the scene of disaster, the cooperation is established and will be maintained with responsible person in charge.

3 The Multi-Agent System

The MAS need to handle several different data representation forms and solve problems quickly. Each agent has a task to accomplish but a result is reached when several agents have found their piece of the information, which is assembled to constitute a result. The task for the agents in our research is to inform, search, match and group information in different parts in the framework, see Figure 1, to be able to combine the extracted information to get more comprehensible and understandable information. The ontologies form a framework for the agents to act upon, selecting the most suitable ontology matching the intention of the agent. The abbreviations are a natural part in the ontologies, however to be treated with specific concern. When extended information is needed, the MAS provide a possibility to search and find it externally, i.e., the web.

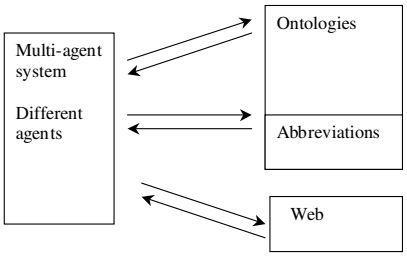


Fig. 1. Simplified architecture of the multi-agent system

The task is performed in episode in which the agents perform a single task. The call to the system is divided into parts, and each part is processed by an agent. For each time the system is invoked with new or more information, the agents must find solutions according to that information. Finding several solutions can granulate the output, which in comparison can support the system to give more accurate output that is more correct according to the situation. In these situations, where searching can be simplified and speeded up using several agents, multi-agent systems are useful [4] [5]. The agents in the multi-agent system must *coordinate* information from several different sources from which they can produce and deliver more complex information in right time to the right place [6], to the appropriate users. The output is the description of the critical situation in international language, as well as, priority of the cases. The information is directives to the rescue services, from which the services can evaluate the situation and determine what kind of service is needed at the scene.

Agents can allow searching for information in several different ontologies and support *reasoning* with the findings [7]. We need this distributed problem solving to search for information in ontologies and priorities in the cases. The agents search for information from several different ontologies and correlate it to match the context of the scene. The international abbreviations are the same, but the interpretation of them, leading to different actions. As for example a drowning situation requires rescue service action in the form of finding, lifting up the human and securing the area preventing other injuries. The ambulance service focuses on Heart-Lung-Rescue, and on giving proper drugs to save the persons life.

4 The Agents

We suggest intelligent agents that learn by being sensible to new or changed environments in combination with using meta-agents. The meta-agents are built on the agents linked to the ontologies, constituting high-level connections between different possible definitions, and matching them to the information collected for the case. The system is launched by the input, which is performed by the agents needed for the task. All the types of agents are invoked to perform their assignment see Figure 2. The "inform" agents work with information and matching the abbreviations to their corresponding words and notify the rescue teams. Some words are explaining seriousness in the case and therefore, some words are used for prioritising.

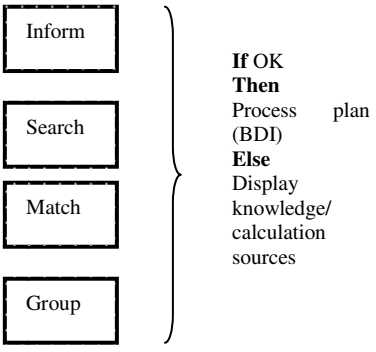


Fig. 2. Inform, search, group and match agents, implemented by BDI model

The "Search agents" search for competence and other teams to work with and "Match" agents perform matching to find right competences for right situations. The agents collect and build up information, which is describing the life critical situations corresponding to the information from the emergency call.

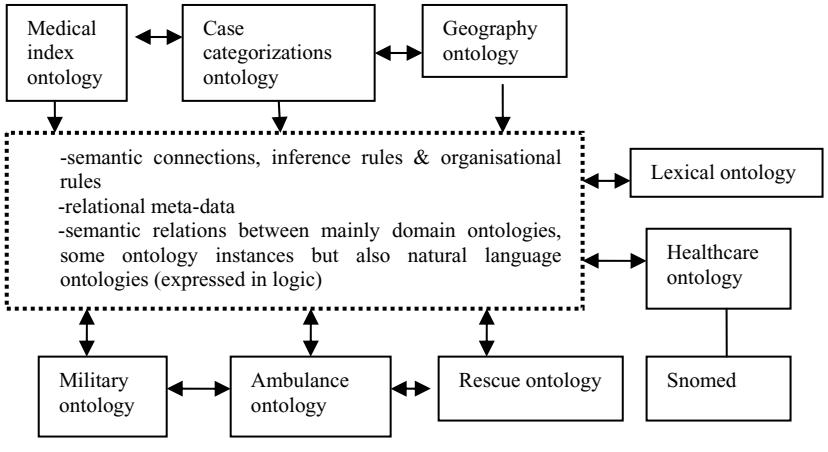


Fig. 3. High-level architecture of ontologies

In emergencies several ontologies are to be used, see Figure 3. The agents' combination of the content of the ontologies determines the situations in which the emergency has arisen. They consist of types, properties and relationships are the knowledge foundation on which the agents perform their work. If a person have a heart stop and do not breathe, Heart-Lung-Rescue actions are to be initiated. From the information, the required services with specific equipment are called. If suitable, the meta-data learn the combination and can be reused for future combinations. These meta-agents are built on the successful agents' performances. Successful agents are the agents that give correct information according to emergency services.

In the *organizational ontology* that is depicted as the military, ambulance, rescue, and healthcare ontology in figure 3, the different parts of the organization are defined and interpreted with the semantic connections including organizational rules. The produced outcome by the agents as they act upon the ontology is: the rescue team that can handle an ambulance case, and the number of operators that should cooperate in handling the accidents. The ontologies are based on measurement of behaviors, such as operator coordination patterns. Another part is structure, which is based on operators own tagging of related information, as used in web 2.0. And example cases of normative behavior are included.

The *medical index ontology* contains of index with instructions about the support that should be given to minimize possible injuries as consequence to the event. For example, the instructions can be how the operator can give Heart-Lung-Rescue support via phone, or which special precautions to take if the person has got diabetes. In the *categorization ontology,* there are structures for how the cases are to be categorized (not to be confused with the ontologies relationships). In the *geographical ontology* there is special support for geographical spots that can have several different names, such as a town park. A town park can have several names, a name on the map, a popular name, and yet another name of the park in relation to its location or surroundings. Alignments tools such as minimal mappings can preferably be used however validations assuring correct connection to the domain need to be performed.

In the categorization ontology the abbreviations are defined. The example of an abbreviation is: H1.11.23. For the trained operator H1.11.23 is: *Human, highest priority, animal bite,* and *unconscious*, which mean that a specific ambulance, with correct equipment and geographical location has to be dispatched. In the healthcare ontology, organisational healthcare rules are defined. *Snomed*[1] [8], a national specialist language for healthcare is connected. The multi-agent system works with ontologies, internally in the system, to find words and produce messages corresponding to the abbreviations, and, externally at the Web, to find geographic location with waved ground. Relations and rules that are not already defined and connected are stored in a database and then further used and updated in suitable ontology. Furthermore, the evaluation of the result of the agents is made in relation to the debriefing session, were also the actions of the agents are collected and evaluated.

5 Conclusions

In this paper, we provided a multi-agent technology to support international handling of life critical cases. The agents act upon input data from the scene of accident, and match information from the ontologies, which are processed to meet the demand. The agents search, match, group and inform the rescue team about the situation, and are implemented by using BDI model. This article is a contribution to the empowering of emergency personnel in life critical contexts performing effect-based operations on an international arena.

[1] Thus, Snomed is not an ontology but a human readable conception system, which (most probably) can form a solid basis for a machine readable ontology.

References

1. http://www.riksdagen.se/webbnav/index.aspx?nid=3281&dok_id=GTB3104d1
2. Lundberg, J., Håkansson, A.: An Approach towards using Agent in Multi-Agent Systems to streamline emergency services. In: Proceedings of 5th International Conference on Information Technology and Applications, Cairns, Queensland, Australia (2008)
3. Woods, W.A.: What's in a link: Foundations for semantic networks. In: Bobrow, D.G., Collins, A.M. (eds.) Representation and Understanding. Academic Press, New York (1975)
4. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge (2008)
5. Bratman, M.E.: Intention, Plans, and Practical Reason. CSLI Publications, Stanford (1987/1999)
6. Apelkrans, M., Håkansson, A.: Information Coordination Using Meta-agents in Information Logistics Processes. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part III. LNCS (LNAI), vol. 5179, pp. 788–798. Springer, Heidelberg (2008)
7. Hartung, R.L., Håkansson, A.: Using Meta-agents to Reason with Multiple Ontologies. In: Nguyen, N.T., Jo, G.-S., Howlett, R.J., Jain, L.C. (eds.) KES-AMSTA 2008. LNCS (LNAI), vol. 4953, pp. 261–270. Springer, Heidelberg (2008)
8. http://www.socialstyrelsen.se/Amnesord/inter_fragor/SnomedCT/specnavigation/Aktuellt/aktuellt.htm
9. Pakhomov, S.: Semi-supervised maximum entropy based approach to acronym and abbreviation normalization in medical texts. In: ACL 2002, Proceedings of the 40th annual meeting on association for computational linguistics (2001)
10. Denning, P.J.: Hastily formed networks. Com. of the ACM 49(4) (April 2006)
11. Molina, M., Blasco, G.: A Multi-agent System for Emergency Decision Support. In: Liu, J., Cheung, Y.-m., Yin, H. (eds.) IDEAL 2003. LNCS, vol. 2690, pp. 43–51. Springer, Heidelberg (2003)
12. Schoenharl, T., Madey, G., Szabó, G., Barabási, A.-L.: WIPER: A multi-agent system for emergency response. In: Proceedings of the 3rd International community on information systems for crises response and management Conference, Newark, NJ, USA (2006)
13. Lundberg, J., Rune, G.: Robust approach towards context dependant information sharing in distributed environments. In: ICEIS 2009, Milan, Italy (2009)

Unifying JIAC Agent Development with AWE

Marco Lützenberger, Tobias Küster, Axel Heßler, and Benjamin Hirsch

DAI-Labor, Technische Universität Berlin
{marco.luetzenberger,tobias.kuester,axel.hessler,
benjamin.hirsch}@dai-labor.de

Abstract. In this paper we describe the Agent World Editor, a tool for designing multi-agent systems and generating executable agent code. The tool also unifies the handling of different agent frameworks through an abstract agent model and an extensible transformation infrastructure. Currently, the tool supports three different agent frameworks of the JIAC family, and we feel confident that the approach holds for other frameworks as well as for the generation of multi-agent systems on heterogenous platforms.

1 Introduction

Over the last decade, Agent Oriented Software Engineering (AOSE) has gained attention as a suitable methodology for providing quality assurance within software development processes. However, there are at least as many agent methodologies as there are agent frameworks, and each has its own drawbacks and advantages. At present, the *DAI-Labor* has three derivatives of its agent framework *JIAC* in use, each one streamlined to a specific field of application:

- JIAC IV [8] has been developed as an agent framework for telecommunication related MAS. The system design is specified in an XML syntax which differentiates three type of agents (platforms, agents and roles) and a set of components like knowledge, services, or plan elements.
- JIAC V [5] was designed to support large agent systems in a scalable way. To this end, the successful features of JIAC IV have been rebuilt with current technologies, which provided an overall improvement in performance and maintainability. The system design is based on the Spring framework to represent the supported agent types (platforms and agents) as well as their components (services or knowledge).
- MicroJIAC [3] is JIAC's lightweight edition for devices with limited resources. The system design is done by an XML based domain model, which provides classes for both supported agent types (platforms and agents) and the framework components (e.g. services and rules), which are used to define specific functionalities or reactive behaviours.

Although the JIAC frameworks — and other agent frameworks as well — feature similarities, there are subtle differences, too. Each framework uses a different model file syntax and provides different libraries.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 220–225, 2009.

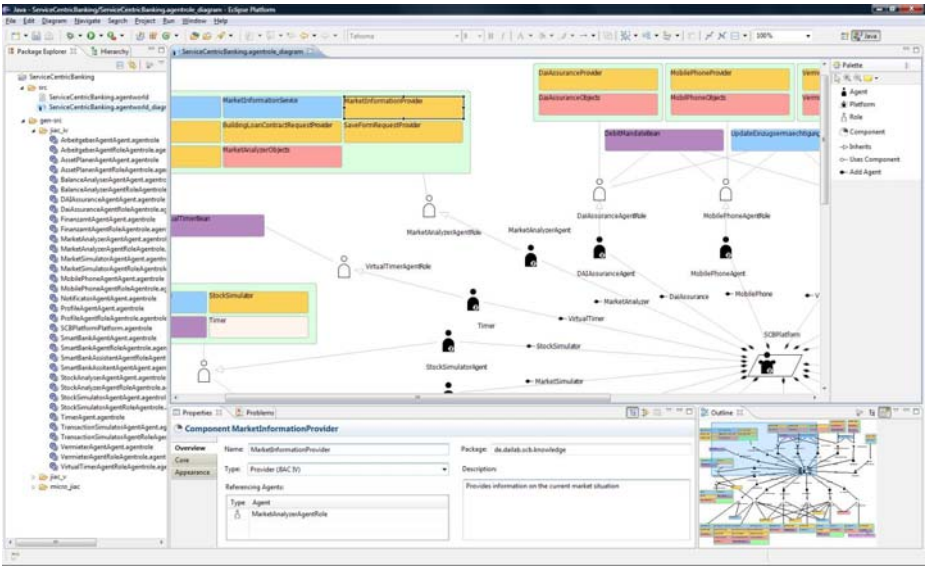


Fig. 1. Agent World Editor showing the *Service Centric Banking* scenario

An abstract conceptualisation of these native library elements enables a comprehensive, framework independent design and constitutes the main idea of the *Agent World Editor*, or *AWE*, which is described in this paper. AWE allows to design multi-agent systems (MAS) and translate the results into different agent frameworks, namely *JIAC IV*, *JIAC V*, and *MicroJIAC*. This paper is based on the diploma thesis of the first author [7].

2 Introducing the Agent World Editor

In AWE, complex multi-agent systems can be conveniently modelled in a single diagram, showing the various agent types, their components and their relationships (see Figure 1). Formal background to these diagrams is a highly generic domain model, which is capable of representing agent system instances of each JIAC derivative. After the MAS has been designed, a framework associated transformation unit can be invoked in order to produce executable agent code and stubs for potentially unimplemented components.

2.1 Generalising JIAC — The AWE Domain Model

The Domain model was designed to encompass the configuration metamodels of at least the three JIAC derivatives, but also to hold for capabilities beyond JIAC. To this end, we accounted for the diverging agent types, i.e. agents, roles and platforms, and countered the remaining challenges by more elaborate mechanisms, which we describe below.

Components: JIAC, like other agent frameworks as well, provides a set of basic agents which are capable of elementary functionality (like communication or environment awareness) and allow individual extension by appending closed and reusable components. The domain model reflects this mechanism with the `Component` class, which is used as an abstraction from the component architectures of specific frameworks.

Library Support: In order to provide support for library elements of multiple agent frameworks, AWE's domain model features the `Concept` class, which is used to provide a framework independent description of their functionality. These conceptualisations are part of AWE's base application and accessed by each installed framework extension by defining an according implementation. Although a comprehensive set of concepts is already provided, AWE's modular architecture allows for easy extension as well. Currently we support development with basic concepts like standard agency, but also provide advanced concepts like an SMS gateway, a calendar feature, and others.

Framework Openness: In order to support frameworks beyond the JIAC scope, each domain model element provides a Key-Value property mechanism. Simple aspects which are not covered by the current model (e.g. teams) can thus be defined. The decision on how these additional attributes are specified as well as their interpretation lies with the developer of the framework extension. An additional object property mechanism enables extensions in an arbitrary granularity.

2.2 Implementation

AWE has been implemented using Eclipse GMF and is based on a plug-in architecture, where each plug-in realises a distinct part of functionality. Support for each agent framework is respectively encapsulated by an individual plug-in, which is loosely coupled to the base application. The standardised structure of these extension plug-ins encourages development of custom solutions, as described in the next section. AWE provides system design in a Drag-and-Drop manner and supports the developer during this process with on-the-fly error validation, an expressive customised visual notation and mechanisms which help to accelerate and simplify recurring tasks. Furthermore, AWE inherits many useful features from GMF, such as unlimited undo and redo, auto-arrange, snap-to-grid, modelling assistance, a graphical outline, picture export, and many more.

2.3 Framework Extensions

Each extension plug-in includes the framework's components, defines a mapping from AWE's abstract concepts to a specific counterpart and defines a code generation for the respective configuration files. The code generation procedure works straight-forward: After the diagram has been validated, the *Agent World* model is exported, iterating over the several agents, roles and platforms, whereupon the several abstract concept elements are substituted with the respective framework-specific implementations provided by the plug-in. Currently, AWE provides extension plug-ins for the three frameworks of the JIAC family.

2.4 Transformation Example

In the following example, a setup consisting of a platform with two agents, a custom Component (`DetectorBean`) and a Concept element (`SMS_Gateway`) is mapped to both MicroJIAC and JIAC V. In both cases an XML file is created according to the syntax used for the respective framework. The component in both cases yields a reference to a Java class (creating a stub, if the class does not yet exist). Most interesting, however, is the mapping of the Concept element: Here, different classes are used for MicroJIAC and JIAC V, namely the library elements implementing this concept as specified in the transformation plug-ins. Figure 2 shows a schematic representation of the transformation.

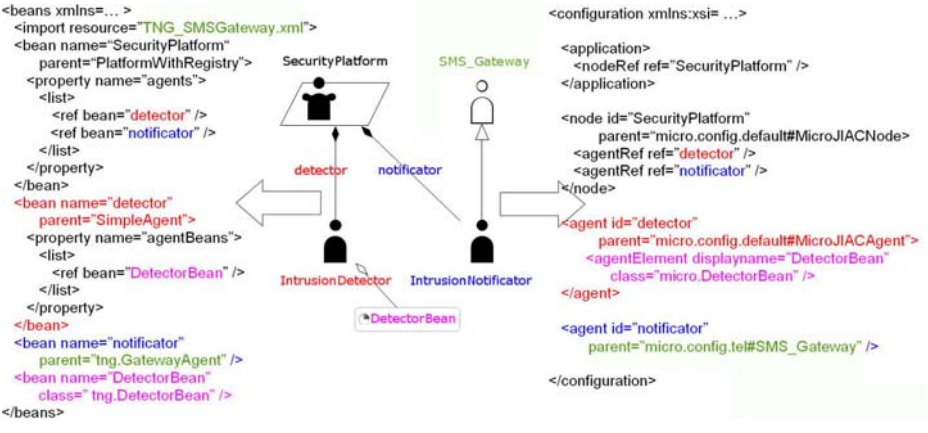


Fig. 2. Code Generation to JIAC V (left) and MicroJIAC (right)

3 Related Work

We have evaluated a number of similar tools, some of them already seeing the 3rd generation [1,9].

The *agentTool* system [10, p. 245–259] is a visual design environment for top-down design of multi-agent systems. In its 3rd edition it is supplemented by a consistency checker, code generation, a metrics calculator and process engineering support. All aspects of an agent-oriented design can be modelled nicely, whereas the code generation feature is more or less an open issue.

The *INGENIAS Development Kit* (IDK) [4] is a visual development tool, targeting the *INGENIAS Agent Framework*. Any model element results in a skeleton that can be extended by the programmer. The IDK has been developed on the basis of an extensible plug-in architecture [4], while the main focus here lies on code generation extensions. Currently, a full translation of the design into executable *JADE* code is provided.

The *JACK Development Environment*(*JDE*) [10, p. 261–277] supports the development of *JACK* based applications. It provides for the creation and manipulation of each JACK component by means of visual engineering and encompasses several other specialised tools. MAS design is accomplished by the *JDE Design Tool* [2], which provides visual engineering on the basis of Drag-and-Drop. Code generation and execution is provided by the JDE as well. The *Compiler Utility Tool* translates the developed diagrams into Java classes and supports both, execution and debugging.

The *Component Agent Framework for domain-Experts* (*CAFnE*) Toolkit [6] provides domain experts a suitable way to easily build multi-agent systems. CAFnE operates on a framework-unspecific domain model, which allows for platform independent design. The toolkit supports visual modelling, code generation, compilation and execution of agent based applications. Currently, a complete transformation module for JACK is available, which converts the platform independent domain model to an executable agent design by using a transformation configuration and a set of transformation rules. The entire mechanism is particularly interesting for this work, since a similar feature is developed as well.

The *JIAC IV Toolipse* [9] is an IDE which facilitates MAS development within the JIAC IV framework. *Toolipse* provides visual engineering of agent applications in terms of diagrams and supports the *JIAC Methodology*. Toolipse encompasses a set of tools, each one for a specific task. Particularly interesting for this work is the *Agent Role Editor* (*ARE*), which is used during the application deployment, in order to define the MAS design. Since ARE has been applied for several years now, various improvement ideas have been considered and inspired this work. Especially its separated representation of agent roles, agents and platforms is considered a major drawback in the system design of multi-agent systems using Toolipse.

4 Conclusion

The Agent World Editor (AWE) is a tool for the visual design of multi-agent systems. It can be used for configuring the various roles, agents, their distribution on several platforms, as well as the components to be used by the agents, e.g. knowledge and capabilities. The aim of the AWE is to be framework independent and highly extensible, such that one agent world model can be used to create setups for different agent frameworks. For this purpose, we introduced the notion of the Concept element, representing an abstract capability of an agent without the need to commit to a specific framework. Upon code generation, Concept elements referenced by an agent in the design are mapped to counterparts in the targeted agent framework. Until now, we have implemented framework extension plug-ins for the JIAC family: JIAC IV, JIAC V and Micro-JIAC. For the near future we are planning to extend this scope by a respective implementation for the JACK framework. Furthermore, AWE will be extended with additional functionality – e.g. a visualisation of the interdependencies of

the several components, especially regarding communication – and it will be integrated into a larger tool suite: While AWE allows for connecting existing components to agents, the development of these components is as yet not supported. Altogether, a versatile set of tools will be combined to an IDE to provide a uniform development application for agent systems.

References

1. The agentTool III Project, http://agenttool.cis.ksu.edu/
2. Agent Oriented Software Pty. Ltd. JACK Intelligent Agents — Design Tool Manual, 5.3 edn. (June 2005), http://www.aosgrp.com/documentation/jack/DesignTool_Manual.pdf
3. Erdene-Ochir, T., Patzlaff, M.: Collecting Gold: MicroJIAC Agents in Multi-Agent Programming Contest. In: Dastani, M.M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 251–255. Springer, Heidelberg (2008)
4. García-Magariño, I., Gómez-Sanz, J.J., Agüera, J.R.P.: A Complete-Computerised Delphi Process with a Multi-agent System. In: Proceedings of the Sixth International Workshop on Programming Multi-Agent Systems, Estoril, Portugal, pp. 187–202 (2008)
5. Hirsch, B., Konnerth, T., Heßler, A.: Merging Agents and Services — the JIAC Agent Platform. In: Bordini, R., Dastani, M., Dix, J., Seghrouchni, A.E.F. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Springer, Berlin (to appear, 2009)
6. Jayatilleke, G., Thangarajah, J., Padgham, L., Winikoff, M.: Component Agent Framework for domain-Experts (CAFnE) Toolkit. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, Hakodate, Japan, May 2006, pp. 1465–1466. ACM, New York (2006)
7. Lützenberger, M.: Development of a Visual Notation and Editor for Unifying the Application Engineering within the JIAC Framework Family. Diploma thesis, Technische Universität Berlin, Berlin, Germany (March 2009)
8. Sesseler, R.: Eine modulare Architektur für dienstbasierte Interaktionen zwischen Agenten. PhD thesis, Technische Universität Berlin (January 2002)
9. Tuguldur, E.-O., Heßler, A., Hirsch, B., Albayrak, S.: Toolipse: An IDE for Development of JIAC Applications. In: Proceedings of PROMAS 2008, Estoril, Portugal (May 2008)
10. Weiß, G., Jakob, R.: Agentenorientierte Softwareentwicklung — Methoden und Tools. Xpert.press. Springer, Berlin/Heidelberg (2005)

Formalizing ARTIS Agent Model Using RT-Maude

Toufik Marir[1], Farid Mokhati[2], and Hassina Seridi-Bouchelaghem[3]

[1] Computer Science Departement, University of Khenchela, Algeria
[2] Computer Science Departement, University of Larbi Ben M'hidi, Algeria
[3] Computer Science Departement, University of Badji Mokhtar, LabGED, Algeria
{Marir.Toufik,Mokhati}@yahoo.fr, Seridi@labged.net

Abstract. Real-time multi-agent systems represent a promising area in the computer science domain. The majority of works realized in such a domain have been developed without tacking into account the formal specification of agent model. Formalizing real time multi-agent systems may offer a solid basis for their verification and validation. We present, in this paper, a formal approach supporting the specification and validation of ARTIS agent using RT-Maude. Based on rewriting logic, RT-Maude is an extension of Maude for specifying and analyzing the real-time and the hybrid systems. Maude is powerful language based on rewriting logic for specifying and analyzing of concurrent systems. Thus, RT-Maude seems to be an appropriate tool for specification, validation and verification of real-time multi-agent systems. The proposed approach essentially allows: (1) translating the description of ARTIS agent, in a RT-Maude specification and, (2) providing a sound description preserving the coherence and supporting their validation process. The generated RT-Maude specifications are validated by simulation. A case study is presented to illustrate the proposed approach.

Keywords: Formal specification, ARTIS Agent, Real-Time Maude, Validation.

1 Introduction

Real-time computer systems represent a big part of the computer systems in our society. In fact, there are many applications under the real-time constraints. Which are generally, complexes, distributed and their principal inconvenient is the difficulty of managing efficiently the applications that work over large data quantity. Thus, the use of real time multi-agent systems seems appropriate and offer more important perspectives. However, the majority of the proposed methodologies for the development of these systems are still in an earlier stage of maturity. Therefore, according to [1] the application of the recent advance in formal method can be helpful in the development of agent-based systems. Consequently, this paper aim to be a first step to formalizing real time multi-agent systems.

The last decade has witnessed the emergence of a number of models and architectures: ANYMAS [2], ARTIS [3], SIMBA [4] supporting the development process of Real Time Multi-Agent Systems (RTMAS). In this paper, the ARTIS model is chosen, thanks to its characteristics. ARTIS model's architecture [3] is one of the interesting answers in the domain of RTMAS development. This architecture is constituted of two levels of agents: ARTIS agent and a set of internal agent (in-agent). The main features of ARTIS

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 226–231, 2009.

agent are: the guaranteed of all critical time requirements by means of off-line scheduling analysis of all reflex actions and existence of toolkit (called InSiDE) that facilitate the design and debugging of ARTIS agent. We note, that only the execution of reflex actions is guaranteed. The temporal restrictions of cognition phase are not guaranteed.

There are many reasons motivating the choice of ARTIS agent as a base for this work. Among these reasons, we can mention: (01) this model guarantees all the hard real-time constraints by means of off-line scheduling; (02) it provides a different layers of abstraction which facilitates the design of complex application; (03) it uses the two well-known methods of Real-Time Artificial Intelligence (anytime method and multiple method) that provide a wide range of choice to the designer depending on future application; (04) it is a model in full evolution where exists many studies and extension based on ARTIS like SIMBA [4] or RT-MESSAGE [5].

Based on rewriting logic, RT-Maude is an extension of Maude for specifying and analyzing the real-time and the hybrid systems. Maude is a powerful language based on rewriting logic for specifying and analyzing of concurrent systems [6]. It has the ability to describe multi-agent systems [7]. Thus, RT-Maude seems to be an appropriate tool for specification, validation and verification of real-time multi-agent systems.

In this paper we propose a formal framework to formalizing a real time agent model baptized ARTIS [3] using RT-Maude [8]. The proposed approach essentially allows: (1) translating the description of ARTIS agent, in a RT-Maude specification and, (2) providing a sound description preserving the coherence and supporting their validation process. The remainder of this paper is structured as follows: In section 2 we illustrate the translation process. Section 3 presents the proposed approach through a concrete case study. Finally, some conclusions are presented in section 4.

2 Translation Process

We developed a formal framework allowing the specification of real-time multi-agent system based on ARTIS agent. As illustrated in figure 1, our framework is composed of several modules: six functional modules (*KNOWLEDGE, BLACKBOARD, KNOWLEDGE-SOURCE, KNOWLEDGE-SOURCE-LIST, MULTI-LEVEL-KNOWLEDGE-SOURCE* and *MULTI-LEVEL-KNOWLEDGE-SOURCE-LIST*), an Object-Oriented module (*IN-AGENT*) and a Timed Object-Oriented module (*ARTIS-AGENT*). Due to limitation space and a considerable size of the proposed framework, we prefer discuss in brief the functional modules and detail the principal modules (*IN-AGENT* and *ARTIS-AGENT*).

The functional module *KNOWLEDGE* is used to define and manipulate the knowledge, which is described using the frame structure. Knowledge is defined essentially by: its title, its kind, its appearance date and its remainder validity time. We have two kinds of knowledge: timed knowledge which is bounded validity time's knowledge and general knowledge which is without validity time restriction.

The functional module *BLACKBOARD* imports the module *KNOWLEDGE* in order to define and manipulate the blackboard structure containing agent's knowledge. This later should be updating during time progressing, especially it should be suppressed if its validity is expired. For each knowledge we associated a list of the InAgent and Knowledge sources that used it, called *InterestedElementList*. This list can be obtained by the function *getInterestedElementList*.

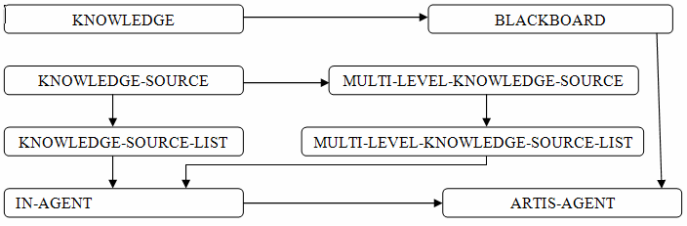


Fig. 1. The framework's architecture

To describe the agent's knowledge source, we define the module *KNOWLEDGE-SOURCE* in which the knowledge source is defined by: an identifier, an execution time and its remainder execution time. This module is used to define the functional module *KNOWLEDGE-SOURCE-LIST* in which we define a list of knowledge source for defining the action ability of an in-agent. On the other hand, the module *KNOWLEDGE-SOURCE* is used to define the functional module *MULTI-LEVEL-KNOWLEDGE-SOURCE*. Multi-level knowledge source is defined as an ordered list of knowledge source level for implementing the anytime algorithm.

The functional module MULTI-LEVEL-KNOWLEDGE-SOURCE-LIST contains the different necessary type declarations and operations used to defining a list of multi level knowledge source which is served to specify perception and cognition agent's ability.

The object-oriented module *IN-AGENT* (fig. 3) describes the form of in-agents. It imports the *MULTI-LEVEL-KNOWLEDGE-SOURCE-LIST* and *KNOWLEDGE-SOURCE-LIST* modules. For a formal description of in-agent, the class *InAgent* is proposed. The definition of this class has the necessary attributes for describing the in-agent. These attributes reflect various parameters included in ARTIS architecture, like the deadline, the period and the priority of in-agent. Among these attributes we quote:

PerceptionAbility, CognitiveAbility and *ActionAbility* which specify, respectively, perception, cognitive and action ability of in-agent; ReactivityParameter which define the reactivity's degree of in-agent; *PerceptionKnowledgeSourceTriggered, CognitiveKnowledgeSourceTriggered* and *ActionKnowledgeSourceTriggered* which specify triggered knowledge source from, respectively, *PerceptionAbility, CognitiveAbility* and *ActionAbility*. The triggered perception and cognitive knowledge source should be scheduling with respect of reactivity's degree of in-agent. The scheduling of knowledge source that means it is inserted to *PerceptionAndCogitiveScheduled-KnowledgeSource*'s list. The attribute *EstimateTimeForAction* is used to estimate the necessary time to execute all triggering action knowledge source before deadline. The attribute *TimerIs* represents a relative timer which estimates the remainder time of active or wait cycle.

CurrentState represent the current state of in-agent. This later can be on of the three possible state: *Activated, Suspended* or *Wait*. On the other hand, the attribute *InAgentKind* of sort *Kind* is used to distinguish critical from not critical agent. For clarity reason, we prefer to illustrate in the rewriting rules only the in-agent's attributes which are necessary to their execution.

The timed object-oriented module *ARTIS-AGENT* is the main module. It imports the *IN-AGENT* and *BLACKBOARD* modules. For a formal description of ARTIS

agents, the class *ArtisAgent* is proposed. The definition of this class has the *InAgentListIs*, *BlackboardContent*, *NbrOfEventIs*, *NbrOfInAgentIs* and *TimerIs* attributes, to contain, in this order, a list of in-agent, a blackboard, a number of events which are not treated, a number of in-agents and a timer.

The ARTIS agent's behavior is specified by two kinds of rules: general rules and specific rules. General rules describe the kernel of agents' behavior which stays unchanged for all ARTIS based applications. The meaning of specific rules is the rules which are specified for each knowledge source of each in-agent. They specify, naturally, the preconditions and post conditions of knowledge source's execution.

In this module we define many messages which are used to construct ARTIS agent's behavior. The most important messages are: *Add*, *Suppress*, *Update* which are used to manipulate blackboard; the messages *Executing, Scheduling* and *Act* are used to indicate the current phase of the in-agent's execution cycle; the message *CurrentPriority* which is used as indicator of the priority of the current execution in-agent; the message *Event* which represents the event perceived by the in-agent. We think that the most important rule is the one that progresses the time, which is called *"tick rule"*. To ensure the uniform progression of time for all timers of in-agents and ARTIS agent, we use a single tick rule. Naturally, in time progression, the ARTIS Agent updates his timer and the validity time of his blackboard. Moreover, ARTIS agent passes the message *TimeProgressToken* to other in-agent and event for update their timers. In this way, the tick rule don't re-execute before the update of all in-agent's and event's timers. Indeed, the update of in-agent's timer (or event's timers) depends on the current state of the in-agent: if the in-agent is in its *activated* state then the in-agent will decrease its timer and the remainder execution time of knowledge source. If the in-agent execute knowledge source from its action ability then it will decrease its timer, its remainder execution time of the knowledge source and its *EstimateTimeForAction*. However, if the in-agent is in *suspended* or *wait* state then it will only decrease its timer. We note that for not critical in-agent we preserve the same cases but we do not decrease the timer of in-agent because not critical in-agent has not period and deadline. For the events we only decrease the appearance date of the event while it is not appeared. Each in-agent can manipulate the blackboard using *add*, *suppress* or *update* messages. In this case, all in-agents and knowledge sources concerned by such a manipulation should be informed. We use the message *Trigger* to trigger all the knowledge source and in-agent exist in the interested elements list of the updated frame.

3 Case Study

This section illustrates the application of the proposed approach on a concrete example taken from [9]. It consists of the formalization of mobile robot based on ARTIS agent architecture. Firstly, we present a brief description of this robot, after that, we validate the developed framework by two possible scenarios.

The formal description of the mobile robot based on ARTIS agent architecture implies all previously defined modules with adding some specific rules relative to its behavior. The interaction between the system and the environment is assured by a set of events: an event to detect the obstacle, and another to indicate periodically the position. To detect the malfunction of robot, two events are used which indicate the

level of power and the connection state. Finally, the request's users transmit via an event which precise the requested task. These events are used to update the system's knowledge. In fact, the blackboard of systems includes the following knowledge: *Position*, *Map*, *LevelOfPower*, *PlannedTask*, *PlannedTrajectory* and *ConnectionState*.

To detect the obstacles; the in-agent should have a topological map of the environment and the current position of the robot. Otherwise, the triggering of the knowledge source *ReadSensor* is canceled.

After the detection of the obstacle, the system triggers the first level of the cognitive multi level knowledge source consist in the first initiative to avoid the obstacle (for example stop movement). Furthermore, the system should update the current map of the environment.

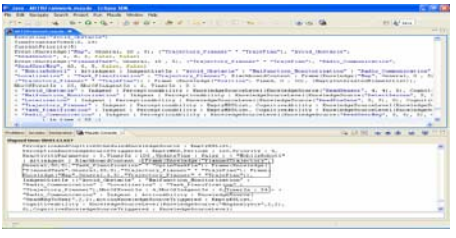


Fig. 2. The initial and final configuration of the first scenario

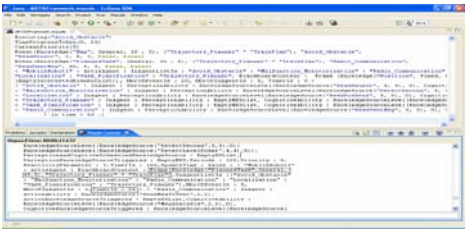


Fig. 3. The initial and the final configuration of the second scenario

Our approach is validated by using two possible scenarios. In the first scenario, there are many events represents the evolution of system's state like: position, level of power, connection state. Among the events presented in the initial configuration, we have an obstacle detected in the second unit of time and user's request appeared at the 40^{th} unit of time. In its initial state, the blackboard includes a description of the environment (Map knowledge) and the current position of the robot. As it is illustrated by the figure (fig.2), the result of the execution of specification is, essentially, the update of the state of blackboard. Indeed, new frames have been added, representing the planned tasks and planned trajectory to serve the request of user. At the same time, the position frame is suppressed because it is timed knowledge and the in-agent charged to identify the new position has not be executed. Furthermore, we remark that the appearance date of the Map frame is six units of time, because the system recognized an obstacle.

We take again the same configuration in the second scenario, but the description of environment is omitted from the blackboard. Evidently, if there is not a topological map, the planned tasks of system can not construct the necessary trajectory to serve the request of user. This situation appears in final configuration where the blackboard includes only a planned task frame (Fig. 3).

4 Conclusion and Future Work

The formalization of the real-time agent represents an important activity during the development process of multi-agents systems. It produces a rigorous description and offers a solid basis for the verification and the validation activities. In this paper we

presented the formalization of one of the well-known real-time agents called ARTIS Agent. We formalized only ARTIS agent which is based on progressive method. Our approach is based on the formal and object-oriented language RT-Maude. It characterizes by the power of description and integrates several tools of verification and validation. In this paper, we only applied the simulation as validation tool.

As future work, we attempt to use this framework for the verification of same properties of real-time agent using the model-checking technique.

References

1. Kefalas, P., Holcombe, M., Eleftherakis, G., Gheorghe, M.: A Formal Method for the Development of Agent-Based Systems. In: Plekhanova, V. (ed.) Intelligent Agent Software Engineering. IDEA Group Publishing, USA (2003)
2. Duvallet, C., Sadeg, B., Cardon, A.: How to build real-time multi agent system with anytime techniques. In: Computers and Their Applications 2000, pp. 325–329 (2000)
3. Botti, V., Carrascosa, C., Julian, V., Soler, J.: The ARTIS Agent Architecture: Modelling Agents in Hard Real-Time Environments. In: Garijo, F.J., Boman, M. (eds.) MAAMAW 1999. LNCS (LNAI), vol. 1647, pp. 63–76. Springer, Heidelberg (1999)
4. Julian, V., Carrascosa, C., Robello, M., Soler, J., Botti, V.: SIMBA: an approach for Real-Time Multi-Agent Systems. In: Escrig, M.T., Toledo, F.J., Golobardes, E. (eds.) CCIA 2002. LNCS (LNAI), vol. 2504. Springer, Heidelberg (2002)
5. Julian, V., Botti, V.: Developing real time multi agents systems. Integrated Computer-Aided Engineering 11, 135–149 (2004)
6. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: aude Manuel (version 2.3) (July 2007), `http://maude.cs.uiuc.edu/maude2-manual/html/index.html`
7. Mokhati, F., Boudiaf, N., Badri, M., Badri, L.: Translating AUML Diagrams into Maude Specifications: A Formal Verification of Agents Interaction Protocols. Journal of Object Technology 6(4) (May-June 2007), `http://www.jot.fm/issues/issue_2007_05/article2/`
8. Carrascosa, C., Bajo, J., Julian, V., Corchado, J.M., Botti, V.: Hybrid multi-agent architecture as a real-time problem-solving model. Expert Syst. Appl. 34(1), 2–17 (2008)
9. Soler, J., Julian, V., Carrascosa, C., Botti, V.: Applying the ARTIS Architecture to Mobile Robot Control. In: IBERAMIA-SBIA 2000, pp. 359–368 (2000)

Implementing Over-Sensing in Heterogeneous Multi-Agent Systems on Top of Artifact-Based Environments

Alessandro Ricci and Michele Piunti

DEIS, Alma Mater Studiorum – Università di Bologna
via Venezia 52, 47023 Cesena, Italy
{a.ricci,michele.piunti}@unibo.it

Abstract. Along with direct interaction models, relying on message-based explicit communication, indirect interaction models can play an important role for designing coordination strategies in Multi-Agent Systems (MAS). Among indirect models, those based on agent *observability* such as *over-sensing* achieve coordination by letting agents observe and possibly reason about (each) others' actions and state. In this paper we describe how over-sensing and, more generally, agent observability mechanism can be used in practice to program MAS in the context of artifact-based environments by exploiting the support provided by CArtAgO technology.

1 Agent Observability in Multi-Agent Programs

In multi-agent systems *observation-based* coordination approaches exploit some kind of *observability* of either the environment or *the agents' themselves* – in particular of their actions and possibly of their state – to achieve coordination among agents. Existing works in agent literature have remarked the features and effectiveness of observation-based interaction and coordination. A main example is the work of Platon and colleagues [7], who introduce the notion of *over-sensing* as a mechanism to let agents observe other agents working in the same environment, providing a first theoretical treatment and discussion of the possible applications. Over-sensing can be seen as a generalisation of *over-hearing*, an indirect communication type that allows agents listening to conversations without the status of addressee [4], in other words to let an agent A to perceive the communication messages exchanged by two or N other agents. Over-hearing has been successfully applied in MAS for monitoring teams of agents and plan recognition [5]. Over-sensing generalises over-hearing by conceptually conceiving observations not (only) related to speech acts but to agent actions/state in general. A comprehensive conceptual account of the role of observation for agent coordination – in particular for cognitive agents – has been provided by Tummolini and colleagues [12], discussing in particular how proper environments supporting observability of agent actions can be effectively exploited in theory to realise different forms of coordination among cognitive agents.

In the context of agent-oriented software engineering (AOSE) – which is the main perspective of this paper – no concrete technology has been devised so far to extensively exploit over-sensing and agent observability in practice. To this end, in this paper

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 232–237, 2009.

we describe a support for agent observability and over-sensing in multi-agent systems composed by agents that interact and work inside *artifact-based* computational environments running on top of CArtAgO.

CArtAgO (Common Artifact infrastructure for Agent Open environment) has been proposed as a general-purpose framework & infrastructure for building shared computational worlds that agents, possibly belonging to heterogeneous agent platforms and written using different agent programming languages, can exploit to work together [9,10,8]. Being based on the A&A (Agents and Artifacts) meta-model [6], CArtAgO's computational environments are modelled as set of distributed workspaces, containing dynamic sets of *artifacts*. From the agent viewpoint, artifacts are first-class entities of agents' world, representing resources and tools that agents can dynamically instantiate, share and use to support individual and collective activities. From the MAS designer viewpoint, artifacts are useful to uniformly design and program those abstractions inside a MAS that are not suitably modelled as agents, and that encapsulate functions to be exploited by individual agents or the overall MAS—for instance mediating and empowering agent interaction and coordination, or wrapping external resources.

CArtAgO provides a concrete computational and programming model for artifacts [9], composed by a set of Java-based API to program artifacts on the one side, and agent API to work inside artifact-based environment on the other side[1].

By integrating an agent programming language or frameworks with CArtAgO, the repertoire of agent actions is extended with a new set of actions for playing inside an artifact-based environment, including actions to join and leave workspaces, create/lookup/dispose artifacts, use artifacts, and observe artifacts without directly using them[2].

Enabling agent observability and over-sensing in artifact-based environments accounts for making it observable the actions that agents perform on artifacts, in particular to execute operations: this point is developed in next section, while in Section 3 we briefly sketch the possible applications of this mechanism.

2 Enabling Agent Observability in CArtAgO

Given the basic CArtAgO model, the observability of agent actions can be enabled by making it observable as artifacts the *agent bodies*, that are those architectural components that – in CArtAgO – are introduced inside the infrastructure to situate agents

[1] The main features of the artifact abstraction are extensively described in [6,10,9]. Briefly, each artifact has a usage interface listing a set of usage interface controls that can be used to trigger and monitor the execution of operations inside the artifact. By executing operations, an artifact can generate observable events (signals) that can be perceived by both by the agent using the artifact and by all those that are focussing (observing) it. Besides observable events, an artifact can have observable properties, whose value (and changes) are automatically perceived by all the observing agents.

[2] The full description of agent API is provided in [9,8]. Briefly, the `use` action is provided to trigger the execution of an operation on an artifact, specifying the usage interface control and parameters; the `focus` action enables the continuous perception of the observable properties of an artifact as percepts and of all the observable events generated by the artifact in executing operations as external events, possibly managed through sensors provided by the agent body.

inside workspaces. For each agent joining a workspace, an agent body is created, containing effectors to act upon workspace artifacts and sensors to perceive workspace observable events, and which governed by the *agent mind* which is in execution externally, on the agent platform side and possibly programmed using different agent languages. So, we achieved observability by *artifact-ising* agent bodies, i.e. by conceiving agent bodies not just as architectural components but as fully-fledged artifacts belonging to the workspace, which can be observed by other agents.

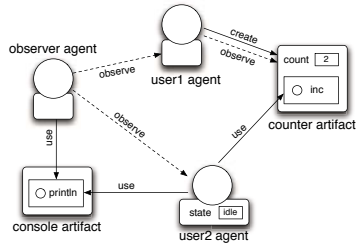
We achieved this by introducing a new built-in kind of artifact called `AgentBodyArtifact`, which is instantiated each time an agent joins a workspace – using the agent name as artifact logic name - and removed as soon as the agent quits. The artifact function is twofold: *(a)* to make agent actions – in particular `use`, `focus`, `observeProperty`, `stopFocussing` observable, by generating proper observable events whenever such actions are performed by the agent. Events are of the kind `doing(`*`What`*`,`*`When`*`)`, where *`What`* can be `use(`*`ArName`*`,`*`Op`*`)` for use actions on the artifact *`ArName`* is the artifact name and *`Op`* is a representation of the operation triggered, and `focus(ArName)`, `stopFocus(ArName)`, `observeProperty(ArName,PropName)` with the obvious meaning; *(b)* to let the agent manifest an observable state, which is represented as observable properties of the agent body artifact. To this end, the repertoire of actions available to agents for working inside workspaces is extended with new ones for manipulating its own observable state, in particular for adding/reading/updating/removing observable properties.

Given this support, an agent can observe another agent's behaviour (or state) just by doing a `focus` action on its agent body artifact. By focussing the body, the agent will perceive all the observable events generated by the artifact and agent observable properties as percepts, in this case related to what the target agent is doing and to the state that the agent has made observable. In so doing, two agents can be mutually aware of each other actions by simply focussing on each other body artifacts.

We clarify the approach through a simple example[3] with three agents, `observer`, `user1`, `user2` working in the same workspace, where `observer` observes and logs the actions done by the other two agents which share and interact with a simple artifact functioning as a counter. All the agents are programmed using the *Jason* agent programming language and platform [2], while the counter artifact is programmed with the Java-based CArtAgO API. Fig. 1 shows the source code of `user1` and `user2`. The agent `user1` simply instantiates a `Counter` artifact named `a-counter` which then will be used by the `user2` agent, and starts observing it, printing a message on the `console` artifact[4] as soon as it perceives a change in the value of `count`, which is an observable property of the counter. The agent `user2` locates and uses the counter, incrementing it few times by acting upon the `inc` usage interface control. This agent also shows an observable property – called `state` – which is used to manifest its working state: first it is `idle`, then, as soon as it starts working with the counter, it is set by the agent to `working` and finally it is set back to `idle` as soon as the agent has completed its work.

[3] The example is part of the CArtAgO technology distribution and can be downloaded at `http://cartago.sourceforge.net`

[4] The `console` artifact is available by default in each workspace and can be used by agents to print messages on standard output.




```
+go_on : true
  <- cartago.makeArtifact("a-counter",
              "counter.Counter",C);
     cartago.focus(C).

+count(X) : true
  <- cartago.use(console,println(
       "new count value observed: ",X)).
```

```
!use_the_counter.
+!use_the_counter : true
  <- cartago.addMyObsProperty(state,"idle").

+go_on : true
  <- cartago.updateMyObsProperty(state,"working");
     !discover_counter(C);
     +cycle(0) ;
     !use_counter(C);
     cartago.updateMyObsProperty(state,"idle").

+!discover_counter(C): true
  <- cartago.lookupArtifact("a-counter",C).
-!discover_counter(C) : true
  <- .wait(1);  !discover_counter(C).

+!use_counter(C) : cycle(N) & N < 4
  <- -cycle(N);
     cartago.use(C,inc,s0);
     cartago.sense(s0,op_exec_completed(_));
     +cycle(N+1) ;
     !use_counter(C).
+!use_counter(C) : cycle(4).
```

Fig. 1. (Top-Left) A schematic representation of the agents and artifacts involved in the example, with in evidence the relationships among the entities. (Bottom-Left) *Jason* source code of the `user1` agent. (Right) *Jason* source code of the `user2` agent.

```
// observer initial goal
!observe_agents.

+!observe_agents : true
  <- !discover_agent("user1");
     !discover_agent("user2").

// plans to discover an agent

+!discover_agent(AgName)
  <- cartago.lookupArtifact(AgName,AgBody);
     cartago.focus(AgBody);
     .send(AgName,tell,go_on).
-!discover_agent(AgentName)
  <- .wait(1); !discover_agent(AgName).
```

```
// plans to react to events related to
// observed agents' actions

+doing(use(ArName,OpName,Params),LocalTime)
      [source(Who)]
  <- cartago.use(console,println("the agent ",Who,
          " did ",OpName," on ",ArName)).

+doing(focus(ArName),LocalTime)
      [source(Who)]
  <- cartago.use(console,println("the agent ",Who,
          " started observing ",ArName)).

+state(X)[artifact(Who)]
  <- cartago.use(console,println("the agent ",Who,
          " state changed to ",X)).
```

Fig. 2. *Jason* source code of the observer agent

```
package counter;
import alice.cartago.*;

public class Counter extends Artifact {

  @OPERATION void init(){
    defineObsProperty("count",0);
  }

  @OPERATION void inc(){
    int count =
        getObsProperty("count").intValue();
    updateObsProperty("count",count+1);
  }
}
```

```
MAS test {
  environment:
    alice.c4jason.CEnvStandalone
  agents:
    observer observer agentArchClass
                      alice.c4jason.CAgentArch;
    user1 count_user1 agentArchClass
                      alice.c4jason.CAgentArch;
    user2 count_user2 agentArchClass
                      alice.c4jason.CAgentArch;
}
```

Fig. 3. *(Left)* The implementation of the counter artifact using the Java based CArtAgO API. The body of artifact operations is implemented by methods annotated by the `@OPERATION` tag. *(Right)* The *Jason* main MAS file, containing the declaration of the set of the booting agents and of the kind of environment where the agent lives.

To add an observable property to its body (artifact) the new `addMyObsProperty` action is used, and `updateMyObsProperty` to change its value.

The key agent of the example is the observer agent, whose goal is to observe the actions performed by `user1` and `user2` and then to log a message on the `console` artifact as soon as either a new agent action is detected or a change of their agent state is perceived (in this case the `state` property of `user2`). Fig. 2 shows the source code of `observer` agent in *Jason*. By means of plans reacting to `+doing(...)` events, the agent prints a message on the `console` artifact, as soon as an event related to a use or focus action performed by the agents is perceived. A message is printed also whenever the belief about a change in the value of the `state` percept is detected (plan `+state(X) [...] <- ...`) – which in this case corresponds to a change to the observable property of the `user2` agent. For completeness, Fig. 3 shows how the counter is implemented using CArtAgO API and the main *Jason* MAS file, declaring the initial set of agents composing the multi-agent systems – `user1`, `user2` and `observer` – and the kind of environment where the agents live – a CArtAgO environment.

Despite its simplicity, the example should be effective in showing how the approach makes it possible quite easily to program agents that choose *dynamically* which agents to observe, possibly observing multiple agents at a time, and that eventually use information resulting from the observation for deliberating and choosing actions to do. It's worth remarking that the same agent can be observed by multiple observer agents, concurrently.

In the example a single not-distributed workspace is used. However the support works analogously also to observe agents not residing on the computational node where the observer is running: in other words, the agent observability support can be exploited in fully distributed work environments. This comes for free by exploiting the native support to distribution provided by CArtAgO: an agent can join and work simultaneously on multiple workspaces, possibly distributed in different CArtAgO nodes.

3 Exploiting Over-Sensing

The basic observability mechanism can be exploited for different purposes in MAS exploiting artifact-based environments.

A first main application for the mechanism is observation-based coordination, as mentioned in Section 1, in the different forms described in [12]. In particular, agents can decide what actions to do and which artifacts to use by observing the behaviour of other agents working in the same environment—which, in this case, do not need to be necessarily aware of the observer agents and to directly communicate with them. By being aware, however, interesting advanced coordination strategies can be enacted, such as *behavioural implicit communication* [3].

Besides coordination, the mechanism can be used for *monitoring* the observable behaviour of agents inside the systems, a feature which is particularly useful in the case of MAS organisations and Electronic Institutions [11] to detect anomalies/violations and, more generally, to enact some kind of security policies which need to trace the interaction between agents and the environment. It's worth noting that in this case the observer agents *are not* mediators of agent actions, such as in the model proposed for situated e-Institutions [1], where agent actions are first sent to a mediator agent (governor), which

is then responsible to execute such actions if they are compatible with the overall norms of the system. In our case, instead, the action is executed directly on the environment (artifacts) and the event related to the action is eventually perceived and processed by observer agents, concurrently.

Finally, monitoring can be used also for profiling purposes, with observer agents keeping track of the timing related to the observed actions of possibly different agents executing core tasks inside the system, so as to reason about the overall performances and eventually identifying dynamically how to adapt the system – for instance changing/adapting the artifacts used by the agents – so as to improve it.

References

1. Arcos, J.L., Noriega, P., Rodríguez-Aguilar, J.A., Sierra, C.: E4mas through electronic institutions. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 184–202. Springer, Heidelberg (2007)
2. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason. John Wiley & Sons, Ltd., Chichester (2007)
3. Castelfranchi, C.: When doing is saying – the theory of behavioral implicit communication. Technical report, Istituto Science e Tecnologie Cognitive (2003). Draft, `http://www.istc.cnr.it/doc/62a_20050131162326t_WhenDoingIsSayingADVANCED.rtf`
4. Dignum, F.P., Vreeswijk, G.A.: Towards a testbed for multi-party dialogues. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 212–230. Springer, Heidelberg (2004)
5. Kaminka, G., Pynadath, D., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan-recognition approach. JAIR 17, 83–135 (2002)
6. Omicini, A., Ricci, A., Viroli, M.: Artifacts in the A&A meta-model for multi-agent systems. Autonomous Agents and Multi-Agent Systems 17(3) (December 2008)
7. Platon, E., Sabouret, N., Honiden, S.: Oversensing with a softbody in the environment. In: In Proc. of Modelling Others from Observation, MOO 2005 (2005)
8. Ricci, A., Piunti, M., Acay, L.D., Bordini, R., Hübner, J., Dastani, M.: Integrating artifact-based environments with heterogeneous agent-programming platforms. In: Proc. of the 7th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008). IFAAMAS, Estoril, Portugal, May 12-16, 2008, pp. 225–232 (2008)
9. Ricci, A., Piunti, M., Viroli, M., Omicini, A.: Environment programming in CArtAgO. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah-Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, vol. 2, pp. 259–288. Springer, Heidelberg (2009)
10. Ricci, A., Viroli, M., Omicini, A.: The A&A programming model & technology for developing agent environments in MAS. In: Dastani, M.M., El Fallah Seghrouchni, A., Ricci, A., Winikoff, M. (eds.) ProMAS 2007. LNCS (LNAI), vol. 4908, pp. 89–106. Springer, Heidelberg (2008)
11. Sierra, C., Rodríguez-Aguilar, J.A., Noriega, P., Esteva, M., Arcos, J.L.: Engineering multi-agent systems as electronic institutions. European Journal for the Informatics Professional V(4) (August 2004)
12. Tummolini, L., Castelfranchi, C., Ricci, A., Viroli, M., Omicini, A.: "Exhibitionists" and "voyeurs" do it better: A shared environment approach for flexible coordination with tacit messages. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 215–231. Springer, Heidelberg (2005)

Requirements and Tools for the Debugging of Multi-Agent Systems

Lawrence Cabac, Till Dörges, Michael Duvigneau, and Daniel Moldt

Department of Computer Science, TGI, University of Hamburg
http://www.informatik.uni-hamburg.de/TGI/

Abstract. Debugging of multi-agent systems (MAS) is hard due to their distributed, concurrent, adaptive, highly interactive, flexible, mobile and heterogeneous nature. We identify three dimensions (*activities*, *scale*, and *coupling*) that span the area of debugging and derive general requirements for a debugging toolset in the multi-agent context. An implementation of a toolset w.r.t. the requirements given for the MAS reference architecture Mulan is presented. This toolset comprises general low level debugging possibilities that are included in the virtual machine (execution engine Renew), specialized Mulan-dependent debugging facilities that enable debugging on higher (agent concepts and independent debugging aspects that rely on publicly available information – i.e. message logs – together with advanced techniques, such as visualization and mining.

Keywords: Debugging, multi-agent systems, requirements, Mulan, toolset.

1 Introduction

Debugging is the process of locating and fixing bugs. Especially the "locating" part is one of the most time consuming and difficult tasks nowadays in software development projects. But before a bug can even be located, its existence needs to be detected. At best, bugs should be detected before the system goes into production. A common approach to check for yet unknown bugs in a project under development is testing [15,12].

In this paper, we concentrate on detecting and locating bugs within multi-agent systems from a software-engineering point of view. Thus the main emphasis lies on multi-agent system-related metaphors for structure (agents) and processes (interactions). The tasks of detecting and locating bugs are already a challenging task in the case of distributed and concurrent systems. Here reproduction of events, control over executing entities and causal dependencies are in many cases beyond the control of the developer. But multi-agent systems are not only concurrent and distributed systems, but also composable and adaptable systems where interfaces of the entities (agents) may change during runtime. This fact imposes another challenge onto testing and debugging as the correctness of a system may vary in different configurations. On the other hand, multi-agent system

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 238–247, 2009.

concepts impose a strong structure on a software architecture. It is advisable to try to benefit from this structure in the process of testing and debugging.

We examine the main characteristics of debugging of multi-agent systems by presenting the requirements for the debugging of multi-agent systems based on the fundamental concepts that are present in such systems as well as in their development processes. Tools are the central means for implementation support as well as sufficient concepts, metaphors, techniques etc. to allow for an effective and efficient productive development setting. For the MULAN/CAPA [7] framework we have developed techniques and tools to reduce the time consumption of the debugging phase.

The paper is structured as follows. Section 2 presents the basic concepts of the debugging process and classifies the related aspects with respect to three dimensions (scale, coupling and activities). On the ground of these dimensions we identifies the resulting requirements. Section 3 introduces our view on multi-agent system development and the matching tools to support detecting and locating bugs. In Section 4 we present related work.

2 Debugging Requirements

The task of detecting and locating bugs involves several *activities* that can be categorized based on their effect within the debugging process. However, activities are not the only dimension to exploit when concerned with debugging a MAS. As a multi-agent system uses metaphors both on micro and macro levels to impose a strong structure on the artifacts that make up a software system, the dimension of *scale* needs to be considered as well. Furthermore, MAS are composable, concurrent, and distributed systems. We group these properties under the dimension of *coupling*. Depending on the kind of bug, the developer has to or needs not to care about the issues imposed by such properties. Thus, debugging activities are spread along this dimension, too.

The most prominent one of the three dimensions is obviously *activities*. Here we find activities that are common to debugging in any paradigm: starting, observing, fixing, compiling and many others. *Scale* on the other hand is a dimension that only gets important in complex systems. While debugging these systems there is an obvious difference whether one examines a method of a class or whether the focus of investigation lies on component level dependencies or even inter-machine coordination. The last dimension *coupling* seems hard to grasp at first, but it should also be obvious that different (maybe also more elaborate) techniques need to be applied when concurrent systems are debugged. With distributed systems also the debugger's (person or tool) limited knowledge of the whole system asks for other approaches.

Debugging multi-agent systems has many aspects in common with debugging normal modular, distributed and concurrent systems. The FIPA standards dictate agent communication via asynchronous message passing. Thus techniques for the debugging of (synchronous) remote procedure calls are not applicable. For example, there is no such thing as a "distributed thread of control". Furthermore, an agent's roles (comparable to interface declarations or class definitions)

are exchangeable during a system's lifetime. No global control or state is present, each agent has incomplete local information, no agent is guaranteed to be able to collect *all* information about a system's current state or behavior. The consequence is that much information available by design in conventional software systems has to either be recovered via observation in multi-agent systems or volunteered by the agents [10]. When information is recovered from observation, techniques such as mining or the exploitation of design artifacts [14] can be applied. The situation becomes slightly better when we add debugging capabilities to a platform instead of injecting a debugging agent as a peer within the MAS, because the platform can inspect *all local* agents and messages while they are executed or sent. Since many platforms provide a framework for agent implementation, the framework can be enriched to observe or manipulate an agent's internals – of course thus violating the agent's autonomy.

From the three dimensions (*scale*, *coupling*, *activities*) given in above the following requirements can be derived:

Display information, on various levels of detail (*observation*). This is the most basic requirement, it covers the whole dimensions of *scale* and *coupling*.

Automation, of the debugging cycle[1] – at least as widely as possible. This relates mainly to the activity *preparation*. The integration of all debugging, testing and development tools helps to satisfy this requirement.

Logging/Tracing. The course of asynchronous events needs to be recorded to investigate the cause of error conditions but also for low scale components logging – as done in conventional debugging approaches – can be of advantage.

Replay. It can be very tedious to manually reproduce an error condition. Therefore a replay mechanism for logged events is desirable.

Distribution. To cover the whole dimension of coupling, debugging tools should be able to remotely connect to every part of the multi-agent system.

Linking, information and artifacts enables the activity "*navigation*". Zooming into an entity (interactive behavior, internal behavior, agent, platform, system) or out of it means moving along the dimension of *scale*. Being able to analyze distributed objects relates to the domain of *coupling*.

Message analysis, is crucial, because agents only communicate via messages. This comprises filtering and mining [1,3] which are essential methods in loosely coupled systems.

Information aggregation, means to be able to condense data as needed. One (simple) example would be to display the number of agents present on a platform rather than each agent individually [1]. Also see the Linking and Visualization requirements.

Visualization, features displaying results from mined data, like for example the communication between a pair of agents or the social network of agents [13].

[1] The debugging cycle defines the process of debugging. This includes starting, observing, finding bugs, determining possible causes, fixing, compiling, restarting, etc. of the system. For more details w.r.t. the MULAN debugging cycle see [5].

Manipulation. All the previous requirements did not change the system being debugged. But capabilities for hot code replacement or data manipulation may speed up the debugging cycle [5]. In contrast to Myers statement that changes should only be done on source code and not on the running system, in multi-agent systems it might be desired or even necessary to do exactly this.

3 Application of Debugging in Mulan

In this section we introduce MULAN, a FIPA-compliant multi-agent platform, and its use to build multi-agent systems within PAOSE, an interpretation of Gaia [17]. Furthermore, we present the facilities that allows us to efficiently debug a multi-agent system. These are the CAPA platform running within the RENEW runtime environment, the MULAN-Viewer and the MULAN-Sniffer.

3.1 Mulan/Capa

MULAN [7] is a reference architecture for multi-agent systems. MULAN models a multi-agent system as a canonical, hierarchical structure of nested, encapsulated elements. A multi-agent system consists of a communication infrastructure, platforms located within the infrastructure, agents residing on platforms, and each agent's internal elements comprising instantiatable protocols (plans), a knowledge base, and internal behaviors (planning, decision components). Platforms offer internal and external communication for agents and harbor special (immobile and privileged) agents that offer AMS and DF functionality. Agents offer and use services and are associated with roles that define responsibilities, abilities, involved interactions and behavioral triggers. Agents communicate with each other using FIPA-ACL.

MULAN in specified in *java reference nets* (see section 3.2). The specification serves as implementation due to the operational semantics of Petri nets. CAPA adds an efficient and elaborated platform implementation to MULAN that supports FIPA-compliant message communication via HTTP.

Due to the fact that multi-agent systems in general and MULAN systems in special are highly structured and loosely coupled, we are able to implement test beds for parts/areas of the system/processes. For decision components (interior processes, DCs) as well as for inter-agent interactions we have developed a technique that allows us to test them in a predefined setting independent of the rest of the system. The designed processes and nets are tested against dummy data or within a crafted multi-agent setup. Such component tests can be run either to detect bugs or to narrow the cause of some error condition down.

3.2 Renew and Java Reference Nets

Java reference nets are a Petri net-based Java extension that allow to conveniently program and run concurrent systems. Java's threading and object communication facilities are replaced by Petri net facilities. Each Petri net graph is

executable code comparable to a class definition. Any statements inscribed to Petri net transitions are executed concurrently, unless restricted by places and arcs. Net instances and Java objects can be used interchangeably. Bidirectional synchronous channels provide a powerful communication mechanism.

The virtual machine that executes the code (Renew [8]) has already several built in features to inspect code at runtime. The Petri net token game can be thought of as a visual debugger that helps to follow the control flow and deeply inspect a system's state. Tokens can be inspected in several ways; as string representation or as UML-like deep inspection (see ellipse highlight in Figure 1) of the object's state. Navigation between encapsulated entities is supported through hyperlink-like functionality, where reference tokens function as links. Renew supports elaborated breakpoints for all relevant entities. However, there are some shortcomings when it comes to the visualization of the overall structure of the (local) system. The next tools avail of the structure of a Mulan-based multi-agent system to present more abstract views of the system.

3.3 Mulan-Viewer

The Mulan-Viewer [2] is particularly strong w.r.t. the requirements *linking information and artifacts*, as it is able to navigate any local or remote multi-agent system (dimension *coupling*) as well as entire platforms or internal items from an agent's knowledge base (dimension *scale*). The main components of the Mulan-Viewer are the platform inspector and the graphical user interface. An arbitrary number of platforms can be inspected both locally and remotely.

The user interface consists of two views: a MAS overview on the left and the detail view on the right (see Figure 1). The hierarchical structure of the multi-agent system is represented as a tree view. The levels of the tree view correspond directly to three of the four levels known from the Mulan model (the outermost level – system infrastructure – is missing). The message transport system agent (MTS) associated with each platform can be seen on the bottom left. If desired, messages can be collected and listed. The detail view allows inspection of chosen elements and provides integration with Renew debugging facilities. It supports direct navigation to Petri net code (as well as net instances) of agents, protocols, decision components and knowledge bases. Additionally, all reactive protocols of an agent are listed so that breakpoints can be set directly from within the tool.

In Figure 1 the running instance of the net *Bank_DC_Account* from agent *bank#7* on platform *poire* has been opened in the detail view (right hand side of the Mulan-Viewer) and can be seen in the background. The superimposed rectangles[2] indicate which elements from the detail view correspond to those in the inspected nets. In this case the string *addResources* shown in the detail view of the Mulan-Viewer is contained by the place in question. The parts marked by superimposed ellipses[2] show how inspection levels can be nested: First the Mulan-Viewer allows for navigation (left hand side of the Mulan-Viewer) to the desired agent and its protocols. Then the Petri nets can be inspected using

[2] Note that the original color of superimposed elements is red.

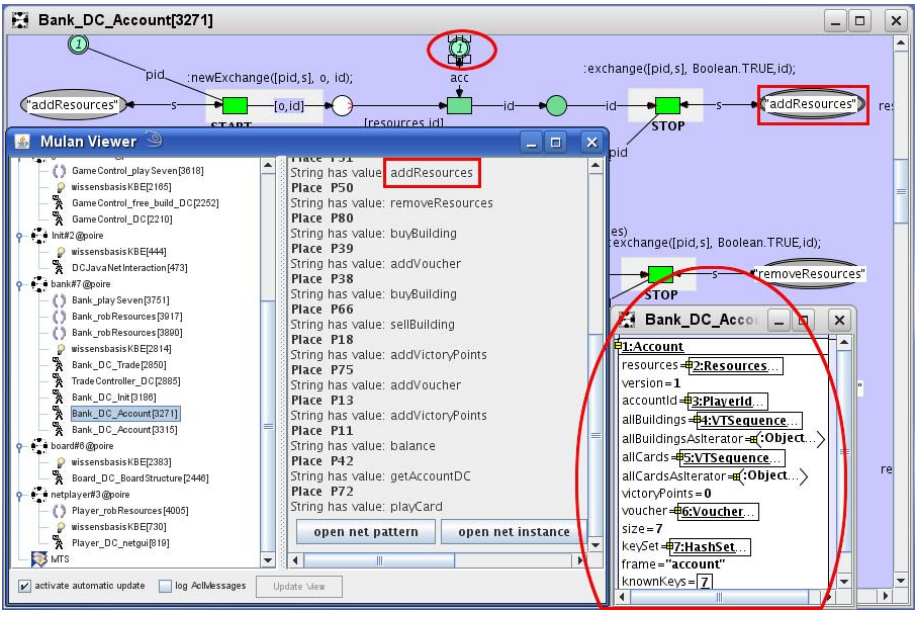


Fig. 1. Mulan-Viewer linking to Renew token game and inspection

Renew. In the example the place on the top contains one token of type *Account*, which is inspected as UML object hierarchy (*token bag*).

Recently, the Mulan-Viewer has been enhanced [5] by powerful (and additional) direct manipulation and control features. These surpass Renew's control and manipulation facilites by utilizing interfaces of Mulan-imposed structure of the multi-agent system. The Mulan-Viewer allows to directly manipulate knowledge base entries (i.e. *hot data replacement)*, allows to start (and stop) arbitrary (e.g. pro-active) protocols, decision components, agents and platforms. Additionally, a "new-protocol-wizard" supports the exchange of faulty protocols (or other nets) by new versions during runtime in a comfortable way.[3]

Thus a cohesion between inspecting, location and fixing of bugs is achieved within the development tool support.

3.4 Mulan-Sniffer

The Mulan-Sniffer [2] was inspired by the JADE sniffer [6]; other related tools and approaches are the ACLAnalyser [1] and the sniffer in MadKit [11]. It uses (agent) interaction protocols (AIP) for visualization and debugging [4,14]. The Mulan-Sniffer focuses on analyzing messages sent by agents in a multi-agent system. Besides from being portable (realized in *Java*) and modular (has complete plugin system), the key features are distribution, logging, analysis, and visualization. The Mulan-Sniffer is able to gather messages from both local and remote

[3] Dynamic loading of nets is already supported by Renew.

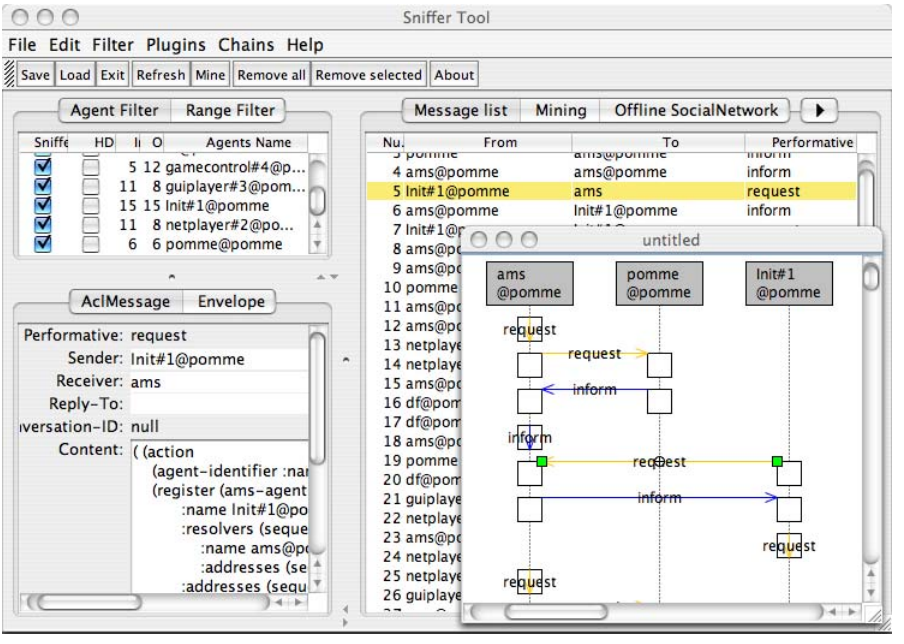


Fig. 2. MULAN-Sniffer UI with generated sequence diagram

platforms. Messages can be selected using stateful or stateless filters. Basic filtering primitives (*from*, *to*, ...) are provided. More sophisticated filters may be added via the plugin system. Offline filtering is also possible. *Mining-chains* can be used to apply arbitrary analysis algorithms to the messages (for examples see [3]). Apart from showing elementary statistics (total number of messages sent, ...) each message can thoroughly be inspected. Moreover sequence diagrams are auto-generated (in function of the filters applied) on the fly. More complex visualizations can – of course – be realized as plugins.

It is interesting to note that the sequence diagrams resulting from visualization can actually be re-used as agent interaction protocol diagrams. Petri net code stubs for agent protocols can be generated from these diagrams [4]. Thus, agent behavior can be defined by observing a running system turning user interaction (manually) into agent behavior or even allowing the agents to use these observations to adapt their own behaviors.

Figure 2 shows the MULAN-Sniffer's main window, while sniffing the messages from a teaching project. The main window is divided in three major areas. The top-left one displays the agents known from sniffing the messages. Here, agents can be selected for simple filtering. The right area shows the *Message list*. The currently selected message is displayed in detail in the bottom left area of the main window. Next to the Message list tab one can select from a couple of viewer plugins loaded already. *Online SocialNetwork* (accessible via the arrow next to *Offline SocialNetwork*) for example allows to visualize the frequency of

message exchange by pairs of agents. Additionally a part of the on-the-fly auto-generated sequence diagram is shown. Selecting a message arrow in the diagram will highlight the corresponding message in the message list and display the content in the message detail view (tabs: *AclMessage* and *Envelope)* and vice versa. The MULAN-Sniffer uses the same interface of the platform as the MULAN-Viewer for the collection of messages.

4 Related Work

Liedekerke and Avouris [16] describe the need for tool support for multi-agent application development. They propose *Developer's Concepual Models* (Perspectives), from which they directly derive views that are manifested as GUI workbench perspectives. The presented system provides visualization covering most of the scale dimension. One main aspect is that the system receives, displays and offers information as an agent of the system. While the approach is modular, generic and abides to the agent-oriented paradigm, there are some drawbacks concerning debugging. Considering that all agents are autonomous entities, it is not guaranteed that operations that query or manipulate data or code are successful. Concerning the dimension of coupling, the architecture seems tempting in the way that the debugging capabilities are naturally remote, concurrent and autonomous. However, direct manipulation of running systems is not discussed.

Ndumu et al. [13] tackle the "notoriously difficult task" of multi-agent system debugging by visualization (of several diverse perspectives of the system) and corroboration. They describe some control features of the system but do not emphasize on a coherent integration of the different tasks of detecting, locating and fixing bugs.

Botia et al. [1] present an elaborated system (ACLAnalyzer) for Jade platforms that focuses on the analysis of ACL messages and the visual presentation of direct or inferred data. Their focus is on the communication level and presents overviews of whole agent organizations. Through this focus on the inferred system organization and the inter-agent communication (interactions) it becomes clear that the tool is mainly for analysis and visualization, while the manipulation features are restricted.

Lynch et al. [10] propose an integrated development environment for multi-agent system. Information about the system or details and control is communicated via messages and the focus lies on the gathering of commonly available information of heterogeneous systems. The agents volunteer the information. Although manipulation and control features are not excluded in the extensible architecture, their support within the basic implementation is limited.

Lam & Barber [9] take a similar approach of gathering information by modifying existing agent code so that the agents report their internal state to a central debugging component.

Myers [12] presents a thorough investigation into testing – mentioning debugging as associated to testing – stresses the fact that the error of fixing of bugs

in runtime code should be avoided. In contrast, we believe that for multi-agent systems (i.e. systems that include adaptive behavior) it is explicitly necessary to manipulate the state directly. Moreover, multi-agent systems actually provide meaningful technical constraints for such manipulations.

Poutakidis et al. [14] focus on the debugging of interactions. They incorporate design artifacts that are developed during the design phase with the Prometheus methodology, i.e. AUML Interaction Protocols, for error detection during runtime. This is a highly specialized approach that is able to find sets of possible erroneous interactions. Errors include mis-sent messages (wrong address) and deadlocks. Such errors are easily and direct detectable in process-oriented multi-agent systems, such as Mulan.

5 Conclusion

This paper investigates the process of debugging multi-agent system from a practical point of view. We base the requirements for debugging of multi-agent systems on three orthogonal dimensions (*scale*, *coupling*, *activities*) that span the field of debugging. Requirements cover gathering, processing and displaying information as well as control and manipulation features of the debugging system.

Table 1. Overview of tools and their capabilities

Observable objects	Viewer	Sniffer	Renew	Component Tests
agents on platforms	++	–	+	–
agent state	+	–	++	–
protocols of agents	++	–	+	++
protocol state	○	–	++	+
transferred messages	+	++	○	+
interactions	–	++	–	++
communication infrastructure	–	+	–	+

We present the concrete toolset to debug Mulan-based multi-agent systems and point out the particular strengths of each tool with respect to the requirements. Table 1 summarizes the features of the debugging toolset for Mulan with respect to the dimension of scale. The Mulan-Viewer focuses on presenting the system structure – including agent internals – and provides control and manipulation features. The Mulan-Sniffer observes the system by gathering, visualizing and mining agent messages externally. Renew provides visual and interactive debugging features for the underlying Petri net code. Together all three form a comprehensive debugging toolset to locate and fix bugs within a Mulan-based multi-agent systems. Component tests allow to detect and locate bugs in a reproducible manner.

References

1. Botía, J.A., Hernansaez, J.M., Skarmeta, F.G.: Towards an approach for debugging mas through the analysis of acl messages. In: Lindemann, G., Denzinger, J., Timm, I.J., Unland, R. (eds.) MATES 2004. LNCS (LNAI), vol. 3187, pp. 301–312. Springer, Heidelberg (2004)
2. Cabac, L., Dörges, T., Rölke, H.: A Monitoring Toolset for Paose. In: van Hee, K.M., Valk, R. (eds.) PETRI NETS 2008. LNCS, vol. 5062, pp. 399–408. Springer, Heidelberg (2008)
3. Cabac, L., Knaak, N., Moldt, D., Rölke, H.: Analysis of multi-agent interactions with process mining techniques. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2006. LNCS (LNAI), vol. 4196, pp. 12–23. Springer, Heidelberg (2006)
4. Cabac, L., Moldt, D.: Formal semantics for AUML agent interaction protocol diagrams. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 47–61. Springer, Heidelberg (2005)
5. Cabac, L., Moldt, D., Schlüter, J.: Adding runtime net manipulation features to mulanviewer. In: AWPN 2008, September 2008. CEUR Workshop Proceedings, vol. 380, pp. 87–92. Universität Rostock (2008)
6. The Sniffer for JADE. Online documentation (January 2008), http://jade.cselt.it/doc/tools/sniffer/index.html
7. Köhler, M., Moldt, D., Rölke, H.: Modelling mobility and mobile agents using nets within nets. In: van der Aalst, W.M.P., Best, E. (eds.) ICATPN 2003. LNCS, vol. 2679, pp. 121–139. Springer, Heidelberg (2003)
8. Kummer, O., Wienberg, F., Duvigneau, M.: Renew– User Guide. University of Hamburg, Faculty of Informatics, Theoretical Foundations Group, Hamburg, release 2.1.1 edn., (July 2008), http://www.renew.de/
9. Lam, D.N., Barber, K.S.: Debugging agent behavior in an implemented agent system. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) PROMAS 2004. LNCS (LNAI), vol. 3346, pp. 104–125. Springer, Heidelberg (2005)
10. Lynch, S., Rajendran, K.: Providing integrated development environments for multi-agent systems. In: Bergmann, R., Lindemann, G., Kirn, S., Pěchouček, M. (eds.) MATES 2008. LNCS (LNAI), vol. 5244, pp. 123–134. Springer, Heidelberg (2008)
11. MadKit (January 2008), http://www.madkit.org
12. Myers, G.J.: The art of software testing, 2nd edn. Wiley & Sons, Hoboken (2004)
13. Ndumu, D.T., Nwana, H.S., Lee, L.C., Collis, J.C.: Visualising and debugging distributed multi-agent systems. In: Agents, pp. 326–333 (1999)
14. Poutakidis, D., Padgham, L., Winikoff, M.: Debugging Multi-agent Systems Using Design Artifacts: The Case of Interaction Protocols. In: Proceedings of AAMAS 2002, pp. 960–967 (2002)
15. Sommerville, I.: Software engineering, 6th edn. Addison-Wesley, Redwood City (1995)
16. Van Liedekerke, M.H., Avouris, N.M.: Debugging multi-agent systems. Information and Software Technology 37, 103–112 (1995)
17. Wooldridge, M., Jennings, N., Kinny, D.: The Gaia methodology for agent-oriented analysis and design. The International Journal of Autonomous Agents and Multi-Agent Systems 3(3), 285–312 (2000)

SONAR*: A Multi-Agent Infrastructure for Active Application Architectures and Inter-organisational Information Systems

Michael Köhler-Bußmeier and Matthias Wester-Ebbinghaus

University of Hamburg, Department of Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
{koehler,wester}@informatik.uni-hamburg.de

Abstract. Whenever the IT infrastructure is managed by different organisational units the software architecture has to balance the conflicting aims of local autonomy and global coherence. Self aware organisational models help to achieve this goal.

In this paper we describe the architecture of a generic organisational agent, called GOPA. This agent is used as the target platform for SONAR-models. SONAR is a formal framework for the specification of multi-agent organisations. SONAR-models are semantically rich enough to compile a GOPA network from the model in an automatic way.

Keywords: Active architectures, multi-agent systems, middleware, organisations, Petri nets, SONAR.

1 Motivation

Organisation-centred design (OCD) models the organisational structure explicitly. OCD has its origins in actor-centred software design and in computational organisation theory [1]. Obviously this approach induces some overhead into the software development processes. Nevertheless these additional modelling costs (in form of resources like time and money) pay out in at least two cases: (1) Whenever software systems are developed for organisation-wide purposes OCD provides the natural metaphors leading to a clear, sometimes even to a clearer design – compared to approaches not aware of organisational structures. This approach is called *business-IT alignment*. (2) Whenever software is not developed as a 'single shot' product but undergoes a constant change then the explicit modelling of the organisational structure has a clear advantage. For example, one major task in 'business re-engineering' is the survey of the organisational relationships and their evaluation – a central part of organisational change management. It would not only be nice but also very interesting from the financial point of view to evaluate and reorganise (if necessary) in an automated manner, i.e. to have an *active architecture*. Therefore reflectivity and self-organisation of business structure can be seen as essential for business information systems.

The situation becomes even more complicated if we do not consider one organisation, but a confederation, like Airbus, an *European Aeronautic Defence*

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 248–257, 2009.

and Space (EADS) company. Citing the company's web page: "Manufacturing, production and sub-assembly of parts for Airbus aircraft are distributed around 16 sites in Europe, with final assembly in Toulouse and Hamburg." Different sites are organisations with some local autonomy which results in a decentralised IT infrastructure with separated authority domains.

This is a typical example where the classical approach of *IT management* does no longer apply since the whole IT infrastructure is spread over autonomous organisational units. E.g. the IT infrastructure in Hamburg is managed by a different organisational unit than the infrastructure of Toulouse and both are quite independent. The local organisational units enjoy a certain *local autonomy*. On the other hand, the confederation as a whole tries to guarantee *global coherence*. This interplay of local autonomy and global coherence characterises the qualitative step from IT management to *IT governance* which heavily relies on self-awareness.

Our formal model SONAR (Self Organising Net ARchitecture) is one step into this direction of self-aware information systems. Firstly, SONAR provides a formal model with a clear semantics. SONAR models come along with a notion of well-formedness that can easily be checked. Secondly, SONAR also provides a clear approach, called SONAR*, to implement the models as a distributed system, namely as a multi-agent system and this implementation (or: deployment) process is carried out as an automated compilation for the most part. Therefore SONAR can be seen as a modelling framework and as a middleware for distributed information systems at the same time.

The paper is structured as follows: Section 2 introduces our concept of holonic organisational networks based on position agents. Section 3 gives a short introduction into SONAR, our formal model of organisations based on Petri nets. The set of possible interaction networks, called teams, is modelled as a delegation net and the set of possible interactions is formalised using multi-party workflow nets. Section 4 describes SONAR* – the generic architecture of the common agent structure. The paper closes with a conclusion and an outlook.

2 Conceptual Approach: Position Network

In this section, we describe the fundamental concepts of our approach before we supply it with a theoretical background and operational specifications in the following sections. Our approach is closely related to a current trend in MAS engineering that takes an organisation-oriented stance and specifically borrows the mechanism of *formalisation* from social organisations. Formalisation in this context refers to the extent, to which expectations on behaviour are explicitly specified, and to the extent, to which these specifications are independent from the particular occupants of social positions [2]. In this respect, rationality resides in a *social structure itself*, not only in the individual participants.

While these are basically *analytical* distinctions in social sciences, applying them to MAS engineering allows for separation of concerns with respect to not only design but also implementation of agents and organisations. In [3], one

can find an overview and a comparison of MAS approaches that all employ the principle of organisational specifications not just resting "in the heads" of domain agents but instead being encapsulated and managed by an explicit middleware layer. Thus, the formal part of an organisation is software technically *reified*. The middleware layer allows arbitrarily heterogeneous agents to participate in the organisation. It is also useful to provide a persistent and coherent system level in an open MAS environment where agents belonging to different stakeholders continuously enter and leave, behaving in possibly unpredicted ways.

Figure 1 illustrates our specific philosophy concerning MAS organisations utilising the middleware approach. We describe a formal organisation in terms of interrelated *organisational positions* (see the following section for details). Compared to other middleware layers, we advocate complete distribution. Instead of introducing one or more middleware managers that watch over the whole organisation (cf. the manager in $\mathcal{S}$-$\mathcal{M}$OISE$^+$ [4]) or at least over considerable parts (cf. institution, scene, transition managers in AMELIE [5]), we associate each position with its own *organisational position agent* (OPA). We provide a more detailed comparison of our approach to other MAS middleware approaches in [6] where we also derive conclusions concerning best fits between different approaches and application contexts.

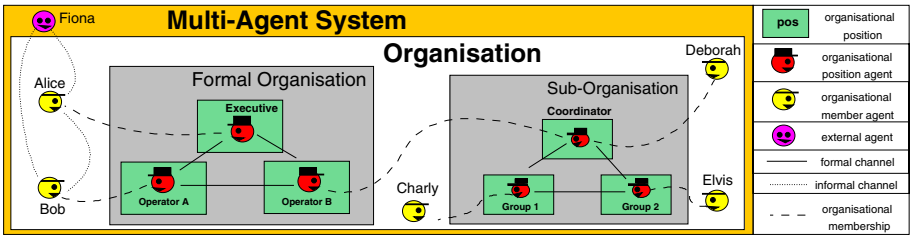


Fig. 1. An Organisation as an OPA/OMA Network

An OPA network embodies a formal organisation. An OPA represents an organisational *artifact* and not a *member/employee* of the organisation. However, each OPA represents a conceptual connection point for an organisational member agent (OMA). An organisation is not complete without OMAs. It depends on domain agents that actually carry out organisational tasks, make decisions where required and thus implement/occupy the formal positions. Note that an OMA can be an artificial as well as a human agent. An OPA both enables and constrains organisational behaviour of its associated OMA. Only via an OPA an OMA can effect the organisation and only in a way that is in conformance with the OPA's specification. In addition, the OPA network as a whole relieves its associated OMAs of a considerable amount of organisational overhead by automating coordination and administration. To put it differently, an OPA offers its OMA a "behaviour corridor" for organisational membership. OMAs might of

course only be partially involved in an organisation and have relationships to multiple other agents than their OPA (even to agents completely external to the organization). From the perspective of the organisation, all other ties than the OPA-OMA link are considered as informal connections.

To conclude, an OPA embodies two conceptual interfaces, the first one between micro and macro level (one OPA versus a network of OPAs) and the second one between formal and informal aspects of an organisation (OPA versus OMA). We can make additional use of this twofold interface. Whenever we have a system of systems setting with multiple scopes or domains of authority (e.g. virtual organisations – like in the Airbus scenario, strategic alliances, organisational fields), we can let an OPA of a given organisation act as a member towards another OPA of another (sub-)organisation as illustrated on the right of Figure 1. For the second OPA, nothing changes. It treats the first OPA as an ordinary member that may or may not maintain further informal relationships. For the first OPA, this move adds just another dimension to its macro level functionality. This basically combines the middleware perspective with a holonic perspective (cf. [7]) and is not as easily to be conceptualised in the context of other middleware approaches that take a less distributed/modular perspective. In Section 4, we address the specific activities of an OPA network in detail.

3 The Underlying Theoretical Model: SONAR

In the following we give a short introduction into our modelling formalism, called SONAR. A detailed discussion of the formalism can be found in [8], its theoretical properties are studied in [9]. A SONAR-model encompasses (i) a data ontology, (ii) a set of interaction models (called *distributed workflow nets*), (iii) a model, that describes the team-based delegation of tasks (called *role/delegation nets*), (iv) a network of organisational positions, and (v) a set of transformation rules (cf. [9,8] for details).

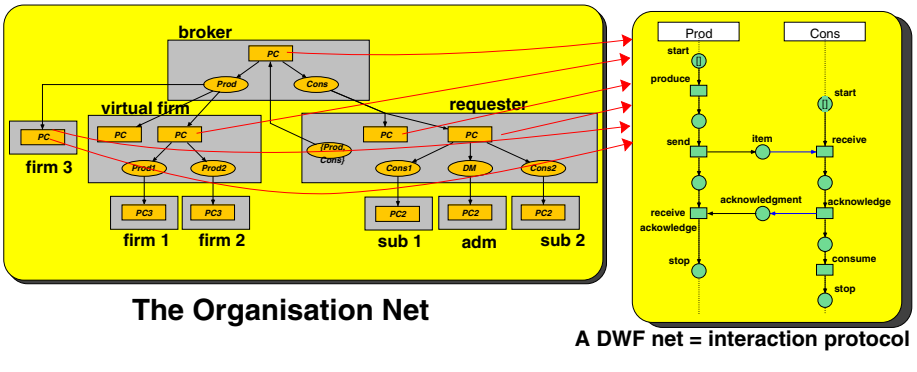


Fig. 2. A SONAR-Model

All these components are illustrated in Figure 2. It describes the relationship between the positions: *broker*, *virtual firm*, *requester*, etc. – and the roles: *Producer*, *Consumer* etc. From a high-level perspective, we have a requester and a supplier of some product. They are loosely coupled and need not even necessarily know each other. Instead, coupling is provided by a broker service. (Note, that for this simplified model brokerage is an easy job, since there is exactly one producer and one consumer. In general, we have several instances for both groups with a broad variety of quality parameters making brokerage a real problem.) From a lower-level, more fine-grained perspective the supplier itself has an inner substructure. It consists of one management level and two subcontractors. The two subcontractors may be legally independent firms that integrate their core competencies in order to form a virtual enterprise. One common separation of concerns is to distinguish fabrication from manufacturing in the development of a product. The coupling between the firms constituting the virtual enterprise is apt to be tighter and more persistent than between requester and supplier at the next higher system level, which provides more of a market-based and on-the-spot connection.

In SONAR a formal organisation is characterised as a network of *organisational positions*. Each position is responsible for the delegation/execution of several tasks and can delegate subtasks to other positions. In our model *Role/Delegation (R/D) nets* [9] are used to describe all the information about task delegation. A R/D net is a Petri net (P, T, F) where each task is modelled by a place p and each task implementation (delegation/execution) is modelled by a transition t. A place p with ${}^\bullet p = \emptyset$ models an *initial task*, while ${}^\bullet p \neq \emptyset$ models a *subtask*. Transitions $t \in T$ with $t^\bullet \neq \emptyset$ are called delegative, transitions with $t^\bullet = \emptyset$ are called executive. Each place p is labelled by a role $R(p)$ and each transition t with a DWF net $D(t)$ (see below). An example R/D net is given on the left side of Figure 2. The positions of our example model in Figure 2 are drawn as grey boxes. It has the positions: *broker*, *virtual firm*, *requester*, etc.

A *distributed workflow net* (DWF net) is a multi-party version of the well-known workflow nets [10] where the parties are called *roles*. Roles are used in DWF nets to abstract from concrete agents. For example, the two roles *Producer* and *Consumer* have the same form of trading interaction no matter which agent is producing or consuming. The right side of Figure 2 shows the DWF net *PC* that describes the interaction between both roles: First the producer executes the activity *produce*, then *sends* the produced *item* to the consumer, who *receives* it. The consumer sends an *acknowledge* to the producer before he *consumes* the item.[1] Technically speaking roles are some kind of type for an agent describing its behaviour. Note, that agents usually implement several roles.

Positions define which entity is responsible for the existing delegation tasks. This relationship is modelled as a set of disjoint subsets of the nodes $P \cup T$ of

[1] To simplify the presentation we have omitted all data-related aspects in our discussion of distributed workflow nets. In SONAR each DWF net uses data object based on the model's ontology.

the R/D net.[2] Each initial task (i.e. the places p with ${}^\bullet p = \emptyset$) are the starting points of organisational activity.

Definition 1. *Let $\mathcal{D}$ be a DWF universe and $\mathcal{R}$ the role universe. A* (formal) organisation net *is the tuple $Org = (N, \mathcal{O}, R, D)$ where:*

1. *$N = (P, T, F)$ is a Petri net with $|p^\bullet| > 0$ for $p \in P$ and $|{}^\bullet t| = 1$ for $t \in T$.*
2. *$\mathcal{O}$ is a partition on the set $P \cup T$. An element $O \in \mathcal{O}$ is called* position.
3. *$R : P \to \mathcal{R}$ is the role assignment.*
4. *$D : T \to \mathcal{D}$ is the DWF net assignment.*

In a well-formed organisation the roles of the DWF net $D(t)$ are consistently related to the roles of the places in the preset and the postset such that no role behaviour is lost or added during the delegation. In a well-formed organisation, termination of the interaction described by a DWF net is guaranteed. Cf. [9] for details.

In general a delegation t comes along with a behaviour refinement. In the example the position *requester* implements the role *Cons* by generating subtasks for the roles *Cons 1*, *DM*, and *Cons 2*. These subtasks are handled by the positions *sub 1*, *adm*, and *sub 2* that implements their role according to the DWF PC_2 (not shown here) which decomposes the behaviour of role *Cons* into the composition of *Cons 1*, *DM*, and *Cons 2*. For the formal definition the interested reader is referred to [9].

If one marks one initial place of an organisation net $Org = (N, \mathcal{O}, R, D)$ with a token, each firing process of the Petri net models a possible delegation process. More precisely, the *token game* is identical to the team formation process (cf. Theorem 4.2 in [9]). It generates a *team net* and a *team DWF*: Teams are modelled as an acyclic R/D nets. More precisely: An R/D net G is called a *team net* if it is a connected causal net (i.e. an acyclic net) with exactly one minimal node: $|{}^\circ G| = 1$. The team DWF is derived from the DWF inscriptions $D(t)$ of the team's maximal nodes $t \in G^\circ$.

As another aspect, SONAR-models are equipped with transformation rules. Transformation rules describe which modifications of the given model are allowed. They are specified as graph rewrite rules [11]. The minimal requirement for rules in SONAR is that they must preserve the correctness of the given organisational model. Cf. [8] for more details. SONAR-transformations are formulated in such a composite way that it is possible to apply them not only to the SONAR model itself but also to the OPA network generated from the model. In the latter case the OPAs perform the transformation within their second-order teamwork (see Section 4).

[2] There is a close connection between organisation nets and the commonly used organisation charts. In fact, organisation charts are a special sub-case of our model. Organisation nets encode the information about delegation structures – similar to charts – and also about the delegation/execution choices of tasks, which is not present in charts. If one fuses all nodes of each position $O \in \mathcal{O}$ into one single node, one obtains a graph which represents the organisation's chart. Obviously, this construction removes all information about the organisational processes.

4 SONAR*: The Generic Position Agent's Architecture

Now that we have obtained a precise picture of what constitutes a formal organisation according to our approach, we can elaborate on the activities of a position network according to Section 2.

We distinguish organisational activities of first- and of second-order. First-order activities target at carrying out business processes to accomplish business tasks. In this case, the organisation is referred to as a static context. In SONAR the first order activities are: Team Formation, Team Plan Formation, Team Plan Execution, and Hierarchic propagation within the holonic structure.

Second-order activities target at evaluation and reorganisation efforts that transform the organisation, which is consequently referred to as a variable.

All OPAs share a common structure which we call the *generic OPA (GOPA)*. An OPA O is an instance of this GOPA that is parametrised by that part of the organisational model that describes O, i.e. its inner structure (subtask and delegation/execution activities) and all the surrounding OPAs. The architecture of the GOPA is shown in Figure 3. In the following we explain its constituting parts. For an in-depth discussion we refer to [12].

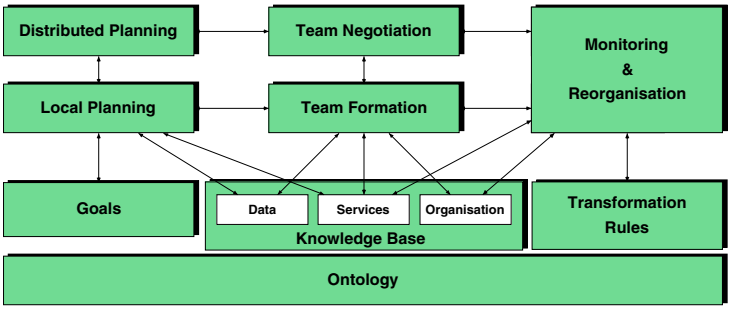


Fig. 3. SONAR*: The Architecture of the Generic OPA

In SONAR* the GOPA specification is formalised using high-level Petri nets. Our first SONAR* prototype uses RENEW [13], a Petri net tool, to simulate the specification directly. To put it short, we have agent nets that in turn utilise protocol nets for agent interactions. Additionally, we employ *GOPA deployment nets* inside protocol nets that guide and control the interaction flow. A screenshot of the running system is shown in Figure 4. It highlights the case of team formation.

Ontology and Knowledge Base. The ontology describes the data types and their relationships. This level uses semantic web technology, like OWL. The data types include basic ones (like integers, strings etc.) and composed ones (like key-value tuples etc). This is the block *Data* in Figure 3. An OPA stores data about e.g. the performance of its neighbour OPAs in it *Knowledge Base*.

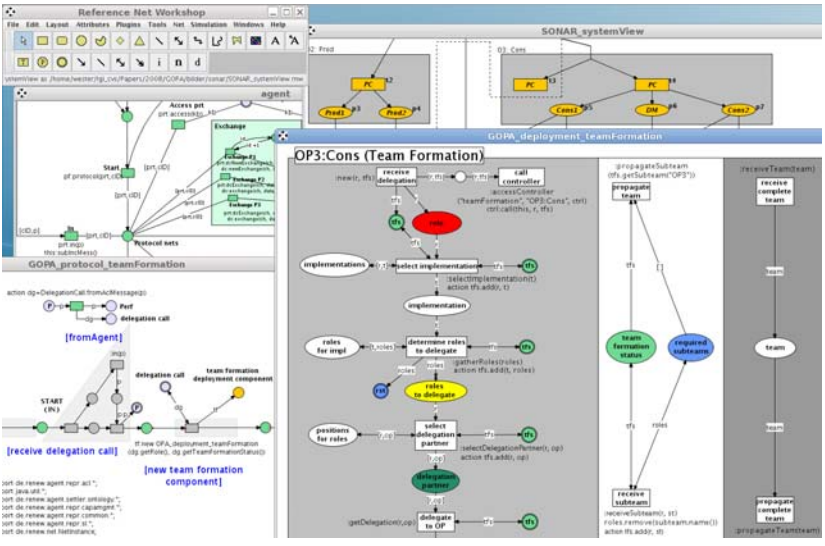


Fig. 4. Screen-Shot of the Executed GOPA Specification

Also DWF nets are described by data types (block *Services*) since the GOPA has to reason about them at run-time when selecting the next action to execute. The third central part describes the organisational parameters of the GOPA (block *organisation*) which includes types as: task, subtask, delegation, execution, in-neighbour, out-neighbour, etc.

Transformation Rules. A special part of the OPAs knowledge is the data base of transformation rules. The knowledge about data, services and organisation can be seen as first-order knowledge used in first-order team-processes. Transformation rules are used in second-order processes, i.e. in teamwork processes that manipulate the first-order data. Typically a second-order process, which carries out a transformation, manipulates the organisational knowledge.

Goals. Each OPA has certain goals. These are not so much concerned with desires like: "Buy a good for the best possible price in the market." since such goals are typically part of the member, the OMA, and not of the OPA. The goals of an OPA are concerned with desires like "I like to minimise the team negotiation overhead in this organisation."

Team Formation. The OPAs use their knowledge about the organisational structure to generate teams for an given task. An SONAR-model provides all the information to describe all the allowed teams. The knowlege about an OPA's neighbour is used to decide which of the possible teams might be the best for a given task. The concrete algorithm has been discussed in Section 4. Additional details of this team formation algorithm can be found in [14].

Local Planning. After a team has been formed all participating OPAs and OMAs articulate their local preferences. In the local planning phase agents deal with

partial plans which also incorporate different choices. This partiality is used in the negotiation phase.

In "normal" interactions only OMAs have an attitude towards different choices within one DWF net. This is different when the team has to carry out a transformation. In this case OPA has strong attitudes which option may improve the organisation best while OMA has no opinion on this organisational topic.

Team Negotiation. After the local preferences have been assigned to the partial plans the team has to negotiate about a compromise. The distributed algorithm for team negotiation is generic and works for each possible team (more details can be found in [8]). It exploits the team structure which is a tree. Each father node in the team-tree investigates the partial plans of it sons. The protocol uses the module for *distributed planning* to compute a compromise among the sons. The result is again a partial plan which is presented to the next level above. In a recursive way this process finally reaches the root of the tree. At the root OPA the preferences of all team members have been taken into account. The best option among the root's partial plan is chosen as the team plan.

Distributed Planning. This modules is used in the negotiation phase in two ways. First it defines how a compromise is computed among the sons' partial plans. Usually it tries to optimise the teams benefit, but also the individual benefit of the team members. Usually both aims contradict each other and the module has to decide which one should be prioritised.

As a second task the distributed planning module has ensure that the locally derived plans do not contradict the organisational constraints. Within the SONAR-model the designer can assign temporal logic constraints to the delegation/execution activities of the organisation net (an issue we have omitted in Section 3). Whenever a local plan contradicts one of the constraints the distributed planner adjusts the preferences in a minimal way to guarantee accordance with the constraints.

Monitoring and Reorganisation. After the negotiation phase the team has computed a team plan which is communicated among all team members. Each OPA is responsible to inform its OMA about the team plan. To enforce OMA's compliance with the team plan OPA monitors OMA's behaviour. In the case that OMA derivates from the team plan an error has occurred. In this case the monitor module is responsible for the error handling, which might a suspension of this OMA or simply message that the action cannot be performed and that another action has to be chosen.

5 Conclusion

In this contribution we presented the generic architecture for an organisational middleware based on multi-agent systems. Our generic OPA-architecture is used as the implementation target of our formal framework SONAR.

In this paper we developed a prototype implementation based on the GOPA specification. In a first attempt we have chosen to exploit the GOPA specification, which is expressed in high-level nets, directly within the Petri net

simulator Renew. As a second, future step we plan to re-implement the specification within the Jade(x)-framework [15] to improve the system's performance.

References

1. Carley, K.M., Gasser, L.: Computational organisation theory. In: Weiß, G. (ed.) Multiagent Systems, pp. 229–330. MIT Press, Cambridge (1999)
2. Scott, W.R.: Organizations: Rational, Natural and Open Systems. Prentice Hall, Englewood Cliffs (2003)
3. Boissier, O., Hübner, J., Sichman, J.S.: Organization oriented programming: From closed to open organizations. In: O'Hare, G.M.P., Ricci, A., O'Grady, M.J., Dikenelli, O. (eds.) ESAW 2006. LNCS (LNAI), vol. 4457, pp. 86–105. Springer, Heidelberg (2007)
4. Hübner, J.F., Sichman, J.S., Boissier, O.: S-moise: A middleware for developing organised multi-agent systems. In: International Workshop on Organizations in Multi-Agent Systems (OOOP 2005), pp. 107–120 (2005)
5. Esteva, M., Rodriguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: An agent-based middleware for electronic institutions. In: Sierra, C., Sonenberg, L., Tambe, M. (eds.) Autonomous Agents and Multi-Agent Systems (AAMAS 2004), pp. 236–243 (2004)
6. Wester-Ebbinghaus, M., Köhler-Bußmeier, M., Moldt, D.: From multi-agent to multi-organization systems: Utilizing middleware approaches. In: Artikis, A., Picard, G., Vercouter, L. (eds.) Engineering Societies in the Agents World, ESAW 2008 (2008)
7. Fischer, K., Schillo, M., Siekmann, J.: Holonic multiagent systems: A foundation for the organization of multiagent systems. In: Mařík, V., McFarlane, D.C., Valckenaers, P. (eds.) HoloMAS 2003. LNCS (LNAI), vol. 2744, pp. 71–80. Springer, Heidelberg (2003)
8. Köhler-Bußmeier, M., Wester-Ebbinghaus, M., Moldt, D.: A formal model for organisational structures behind process-aware information systems. Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems 5460, 98–114 (2009)
9. Köhler, M.: A formal model of multi-agent organisations. Fundamenta Informaticae 79(3-4), 415–430 (2007)
10. Aalst, W.v.d.: Verification of workflow nets. In: Azéma, P., Balbo, G. (eds.) ICATPN 1997. LNCS, vol. 1248, pp. 407–426. Springer, Heidelberg (1997)
11. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of algebraic graph transformation. Springer, Heidelberg (2006)
12. Köhler-Bußmeier, M.: SONAR: Eine sozialtheoretisch fundierte Multiagentensystemarchitektur. In: Lüde, R.v., Moldt, D., Valk, R. (eds.) Selbstorganisation und Governance in künstlichen und sozialen Systemen. Lit Verlag, Münster (2009)
13. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – the Reference Net Workshop, Release 2.1. (1999-2009), http://www.renew.de/
14. Köhler-Bußmeier, M., Wester-Ebbinghaus, M.: Automatic generation of distributed team formation algorithms from organizational models. In: Hubner, J.F., et al. (eds.) COIN 2008. LNCS (LNAI), vol. 5428, pp. 64–79. Springer, Heidelberg (2009)
15. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A short overview. In: Net. ObjectDays 2004, pp. 195–207 (2004)

An Architecture for Simulating Internet-of-Services Economies

Stefan König[1], Isaac Pinyol[2], Daniel Villatoro[2], Jordi Sabater-Mir[2], and Torsten Eymann[1]

[1] University of Bayreuth
Chair of Information Systems Management
Bayreuth, Germany
stefan.koenig@uni-bayreuth.de
[2] Artificial Intelligence Research Institute (IIIA)
Spanish National Research Council (CSIC) Bellatera, Barcelona, Spain

Abstract. The Internet-of-Services describes a general paradigm of distributed computing with transparent service selection in a shared infrastructure. One particular question to be solved is how to match service supply and demand dynamically, while information is asymmetrically distributed between buyers and sellers (represented by agents). As buyers can not investigate the computation service before use, sellers can behave strategically. Including trust-enhancing market concepts or reputation mechanisms can help to lower this information gap. Researching into this effect requires the setup of simulation environments, that allow to change policies (e.g. market structure or reputation parameters). In this paper, we present an architecture of a simulation environment integrating electronic institutions from multi-agent research to simulate Internet-of-Services systems.

1 Introduction

Businesses have to encounter several different challenges, when it comes to using Information Technology (IT). The increasing dynamism of markets leads to a continuous need for IT-Business-Alignment and the control of IT investments and resources. The Internet-of-Services (IoS) describes a general computational paradigm, which allows companies to procure computational resources externally and thus to save both, internal capital expenditures and operational costs. For the provider of Internet-of-Services, the business model lies on the economies of scales.

From a technical point of view, Internet-of-Services virtualizes physical resources to logical units, which can be assigned to different users. Thus, using resources in parallel becomes possible, leading to overall better utilization and the execution of computationally intensive jobs within shorter time. One important characteristic is the distributed and perhaps redundant provision of storage, processing power, or more abstract services that extend over different organizations [1,2]. The heterogeneity of services and resources is opaque to the end user.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 258–267, 2009.

1.1 Markets to Allocate Resources and Services

An efficient allocation mechanism between service demand and supply is needed to get such an environment running – a market. But introducing markets will lead to other problems, e.g. asymmetrically distributed information between service providers and consumers. Service providers usually have more information about quality or availability of the services, than their potential users (consumers). This case of asymmetrically distributed information usually leads to suboptimal results due to the uncertainty on the consumer side, and thus to an inferior usage of the service environment in total. Symmetrically, consumers have more information about their liquidity. In addition, both transaction participants deal with uncertainty caused by environmental factors (e.g. network failures). Effects, which are observed in physical markets, can be found in service environments as well [3,4]. Without regulation and coordination mechanisms, the Internet-of-Services can thus suffer from low quality of service (QoS), like in the proverbial "market of lemons" [5].

Because of these economic issues, policies as kind of rules need to be defined to overcome these shortcomings. In this sense, finding strategies and defining policies to ensure certain QoS in the Internet-of-Services must rely on (1) predefined negotiation protocols, following institutional approaches and (2) subjective distributed mechanism, following social approaches (for instance, by the use of reputation and trust mechanisms). Because of the context-dependence of both conditions, simulating tools seem a good solution to test theoretical hypotheses if they allow to change these policies. Therefore we recommend to make use of electronic institutions.

1.2 The Usage of Electronic Institutions in the IoS

On the one hand, negotiation protocols must provide general policies that all participants have to follow. On the other hand, since not all policies are enforceable in all system physiologies, trust and reputation mechanisms, similar to those used in e-Commerce applications, arise as a good distributed solution. Especially systems involving different organizations or open systems hamper the introduction of consistent policies [4]. However, our understanding of trust extends the prevalent technical-oriented views in service environments. Secure communication and digital certificates are necessary, but not sufficient to generate trust both to the system and to other participants. However, trust has also to be seen as a social paradigm, which can be built dynamically between agents with regard to its past behaviour. This social approach offers as well an interaction control that escapes from the security approach and becomes critical for achieving a well-fare market with asymmetrically distributed information.

The use of eI in the simulation tool provides us several advantages in the context of simulating policies for the IoS:

– A completely integrated and widely used framework to graphically design and deploy eI through EIDE[1]. This provides us with tools to easily design

[1] http://e-institutor.iiia.csic.es

negotiation protocols, improving their conceptualization and possible modifications. Further, it provides tools for monitoring at run-time agent's performances in the negotiation protocols.
- An extensible agent architecture (*E-Agent*) to allow software agents to participate in eI. Then, software agents using complex architectures like BDI (*Belief, Desire, Intention*) can easily participate in eI.
- Reputation and trust models management. The E-Agent architecture ensures a completely interoperability of agents using different reputation models in the same eI [6].
- Direct applicability for real applications. In fact, EIDE provides a set of tools to allow humans to participate in eI. Therefore, after an experiment has been simulated using the platform, exactly the same design of eI could be used as negotiation protocols involving humans and virtual agents.

The paper is structured as follows: In the next section we state some related work. Afterwards, in section 3 we expose a brief background on electronic institutions concepts and tools, together with the conceptualization of Internet-of-Services economies as electronic institutions. Also in this section we explain how eIs deal with reputation mechanisms. In section 4 we describe the abstract architecture and put the simulation to work by presenting an instantiation of a possible simulation and showing some results. Finally, we conclude the paper and present the future work in section 5.

2 Related Work

Using intelligent agents for trading resources in the Internet-of-Services is not really new. Foster et al. [7] state that both, Grid computing and software agents, are about the design of distributed systems. Whereas the Grid community focused on the development of the "infrastructure and tools for secure and reliable resource sharing within dynamic and geographically distributed virtual organizations (VOs)" [7, p. 1], the agent community focused on the development of intelligent agents being able to act in uncertain and dynamic environments.

In order to model different characteristics of services and resources, we use a two-layer market, which has been evaluated through a prototype in the CATNETS project (see figure 1). The project defines a market for services (SaaS-market) and infrastructures (IaaS-market). The main differences of this paper to this approach are the assumptions regarding agents' behaviour: the agents are not assumed to act cooperatively.

A set of additional simulation toolkits for distributed systems has been designed. One of the most promising ones is the OptorSim toolkit [8]. However, this system is not further developed as of 2006 and therefore lacks user support and adoption to future network settings. GridSim [9] presents a quite comprehensive simulation framework; it focuses strongly on Grid Computing applications, thus stressing technical details such as scheduling or generation of virtual organizations and advance reservation of resources. Instead, our aim is to build an architecture combined with some existing tools that enables an easy-to-deploy

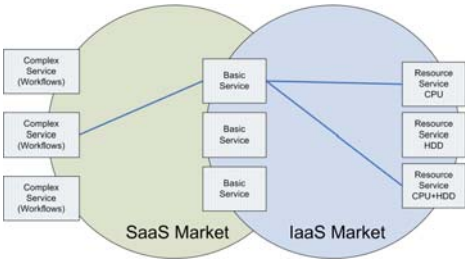


Fig. 1. SaaS and IaaS Markets

simulation environment, even for researchers that are not really familiared with programming simulations.

In the next section we give a brief description of the main concepts and characteristics of electronic institution.

3 Electronic Institutions for Internet-of-Services Economies

3.1 Electronic Institutions: Basic Concepts and Tools

In everyday life, we, as individuals, deal with many other people in order to achieve our goals. Many of these interactions are regulated by an institution that takes care we are following a set of norms and protocols. The concept of *electronic institution* [10] is inspired from these human institutions. Open multiagent systems are composed of autonomous entities that interact to achieve individual or collective goals. The behaviour of these entities cannot be guaranteed. Therefore, and similar to what happens in human societies, we need mechanisms to guarantee the well working of the system in spite of local behaviours. The use of an electronic institution that regulates the behaviour of agents is one of this mechanisms, and can be complemented by other mechanisms like, for example, the use of reputation. You can find a running example [11] of the usage of electronic institutions for the usage of simulating the behavior of humans in hybrid experiments.

Summarizing, we identify the following main components regarding eI: *Agents* are the players in an electronic institution, interacting by the exchange of illocutions, whereas *roles* are defined as standard patterns of behaviour. EIs establish the acceptable speech acts by defining the ontology and the common language for communication and knowledge representation, which are bundled in what is called a *Dialogical Framework*. Interactions between agents are articulated through agent group meetings (which are called *scenes*) with a well-defined protocol. Protocols in a scene are considered to be the specification of the possible dialogues the participating agents may have. Scenes can be connected, composing a work flow, in a so-called performative structure. The specification of a *performative structure* contains a description of how agents can legally move from

scene to scene. The purpose of the normative rules is to modify the behaviour of agents by imposing obligations or prohibitions. Institutional agents are committed to undertake the required actions so as to ensure that non-institutional agents abide by *institutional rules*.

The *Electronic Institutions Development Environment* (EIDE), an integrated development environment for Electronic Institutions, is a set of tools developed at the IIIA-CSIC aimed at engineering multiagent systems as electronic institutions. ISLANDER [12] is the graphical tool to specify eI, and AMELI [13] is the agent-based middleware to run eI. In the next subsection we use these concepts to specify an eI that captures the nature of Internet-of-Services economies in the context of a multiagent system.

3.2 Using Electronic Institutions in the Context of Service Economies

To use an eI, it is necessary to define agents and roles, a dialogical framework, scenes, a performative structure and normative rules in accordance to the identified service markets (see figure 1) and agent roles:

Complex Service agents buy high level services on the service market. *Resource agents*, on the other side, sell their resources (e.g. disk space or CPU power) on the resource market. *Basic Service agents* are the only agent type acting on both markets. They offer their services on the service market and buy and recombine resource services on the resource market. Beside these agents we have to define staff agents, which offer auction services. The participating agents can optional exhibit the role of an staff agent. That means, they can sell their own products or demands. Ontological problems that occur in real distributed systems are not addressed by this paper.

A bit more interesting seems to be the performative structure of our Internet-of-Services use case. Figure 2 illustrates the definition made with the ISLANDER tool [12].

We will now consider the most important scenes. Some of them have just "technical" reasons. After the mandatory initial state all agents have to register at the eI. Additionally, Resource agents are able to register their resources. Once the resource agent has successfully sold its resource for a certain time it can decide how to proceed: leaving the e-Institution (as kind of marketplace) or providing another resource service again. With the latter case, we are able to simulate one kind of cheating behaviour of single agents. The Complex Service agents' possibilities are analogous to the resource agents: after buying a service they can choose between leaving the system or buying another service. A bit more complicated is the route of the Basic Service agents through the different scenes: a Basic Service agent has to buy resources on the resource market. The resource auction is implemented through the scene *AuctionRM*. Only after succeeding on the resource market the Basic Service agent is able to provide its basic service on the service market. To do this, it has to join the *ItemRegisterSM* scene. After finishing the service auction there, the Basic Service agent can decide if it leaves the e-Institution or runs through this cycle again.

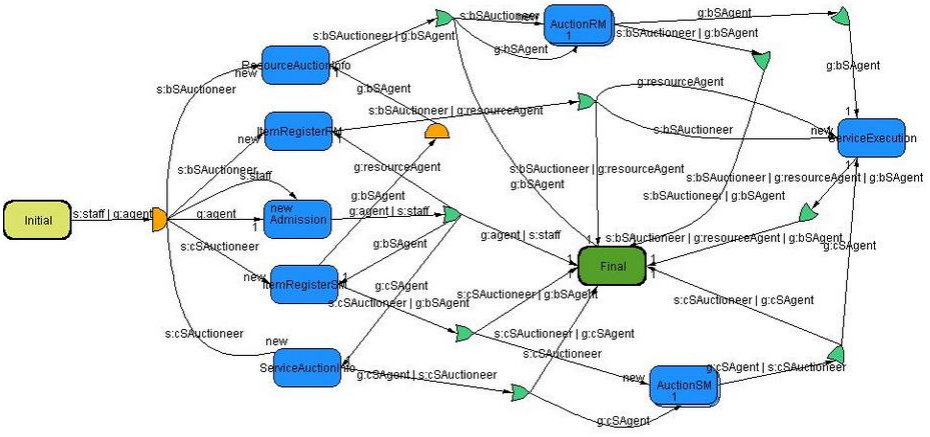


Fig. 2. Service Economy Model implemented as e-Institution. In the scenes *AuctionRM* and *AuctionSM* agents participate in auctions in the resource market and the service market respectively. Previously, all agents and products have been registered in some of the other scenes.

In the *ServiceExecution* scene all agent types meet up after performing successful transactions on the markets. They try to execute the service invocation. If the resource agent for example decides to cheat and not to execute the service, it notifies the other agents in this scene.

The way staff agents take through the performative structure is like follows: they create new instances of their scenes, e.g. the *Service Auctioneer* (and analogous the *Resource Auctioneer*) creates an instance of the *AuctionSM* (i.e. the service market). After that it just waits for products to sell. Note: Currently we have just implemented an electronic Institution market definition for auctions. Changing this to some other (smarter) negotiation protocols will be future work.

3.3 Reputation Mechanisms and Electronic Institutions

As we mention in the introduction, a social vision of trust provides another level of interaction control. The security approach guarantees privacy, authenticity and integrity of information, and the institutional approach guarantees protocols of interaction and performative structures. From a social approach it is desirable that the same customers, as autonomous entities, have access to some social information to dynamically build their trust towards other members of the society. One source of social information is *reputation*.

From a computational point of view, trust and reputation models have attracted increasing interest in the field of multiagent systems. The models that appeared in the literature mainly follow two different and well-distinguished approaches. On the one hand, centralized reputation models consider trust or reputation as a global and public property of the agent that everybody can observe. In this sense, reputation values are kept by a central authority that is

accessible for all participants. These kinds of models are widely used in online markets like eBay or Amazon. On the other hand, distributed models consider trust or reputation as a subjective property of the agent. So, each agent has its own vision of the society. Through its experience, observation and maybe communications from other members each agent creates its own opinion, that can be shared, or not.

- **Centralized Models in eI:** These models need a considerable amount of information to be reliable. Therefore, the most sensible thing is that they could be denoted as a service of an eI. Different reputation models are provided through the usage of the eI.
- **Distributed Models in eI:** In this case, each agent has its own reputation model, and because of that, another problem arises: the interoperability between agents using different reputation models. This problem is partially solved in [6], by proposing a common ontology on reputation concepts that all agents have to share and use when communicating with other agents.

4 Integrating eI into an Internet-of-Services Simulator

This section briefly describes the overall architecture. The e-Institutions are used as a kind of marketplace to trade resources or services. In such a marketplace different negotiation protocols can be implemented. On the one hand resources and services (auctions) or demand (reverse auctions) can be offered or both of them (double sided markets like Continuous Double Auctions or through bargaining). On the other hand, agents must be able to use a central or distributed reputation mechanism to consider the reputation impacts on service markets.

While the last section focused on the e-Institutions layer, we will now consider the functionalities of the simulator core and the agent architecture.

In an advanced stage of the prototype implementation different competing marketplaces could be thinkable. Most of the agent process phases (i.e. Selection, Negotiation, Execution and Enforcement phase) are fulfilled by the eI functionality. Only the Execution phase, as a very specific one, has to be added (simulated or real). Further, different scenarios can be simulated, for example some agents defecting all the time, with a certain probability or the smartest one, agents deciding based on their strategy if they cooperate or defect, to allow for strategic agent behaviour.

4.1 Simulator Core Functionalities

A Scenario Generator is able to generate a grid network. The Application Layer Network is composed of different interconnected nodes. Each node can host different agents, which represent a certain resource or resource bundle (Resource agent) or a higher abstracted service (Basic Service and Complex Service agents). The network determines the time delay between sending a message and receiving it. So, a service with a very short-term time restriction might fail due to the message time delay between the corresponding nodes.

The network representing an IoS is defined by a connected non-oriented graph, represented by a set of network sites $S = \{1, .., n\}$ and a set of links between the sites $L = \{\langle i_1, j_1\rangle, .., \langle i_m, j_m\rangle\}$. In addition, a failure probability f_{S_i} is defined for each node. When a failure occurs during simulation, the node is not able to answer any request or routing further messages. Which site will fail in each time tick, is chosen randomly. Furthermore, on each node a set of the three different agent types, Complex Service Agents (CSA), Basic Service Agents (BSA) and Resource Agents (RA) is initialized. For each site the number of economic agents is $|CSA_{S_i}| \geq 0, |BSA_{S_i}| \geq 0, |RA_{S_i}| \geq 0$. A node without any associated economic agent is a router. Each link $\langle i, j\rangle$ between two nodes has a certain bandwidth. A higher bandwidth leads to an increased data transfer. In our simulation model, the bandwidth is biased, which means that a link is defined or not. Nodes, which are not directly linked can be addressed through a routing table that is calculated by a common routing algorithm, the Dijkstra algorithm [14].

Further, a network connection between nodes might break down. Agents are not longer connected to each other. This might also cause problems during service execution. In addition to these functionalities, the simulator provides a resource locking model. If resources are sold, they are locked and can not provide their capacity to other agents. This does not avoid the cheating behaviour that agents might sell a service or resource twice. But during the execution phase, the resource can be locked only once, such that one service can not be fulfilled by the agent.

4.2 Experimenting with the Simulation Platform

In the following, we present a simple instantiation of the simulation platform to illustrate the potential of this tool. Even when the objective of this paper is not to test any experimental hypothesis we want to show the flexibility that our simulator offers when testing Internet-of-Services economies.

In this case, taking as a base the eI specified in figure 2 we decided to use an English Auction protocol for both, resource and service markets negotiation. ISLANDER offers graphical tools to specify such protocols. Thus, the modification or even a complete exchange of interaction protocols is quite simple and furthermore, can be used in both the simulation platform or real environments where eI is used. The simulation process works as mentioned in subsection 3.2. The entry point to the simulation is the CSA. The CSA has to fulfill an external generated demand. In our simulation the demand is generated by an uniformly distributed interval span. The kind of Basic Service that the CSA has to buy is also given by demand generation. The BSA on the other hand has to compose different resources by a certain combination. The demanded resource bundle can differ between the BSA-types. After the agents have found an agreement, the settlement phase is simulated through the exchange of messages, which are sent from the invoking agent to the sold agent, together with their corresponding answer, if the invocation has been successful.

Figure 3 shows the market price on the service and on the resource market. This simulation run has been conducted with ten nodes, hosting eleven CSAs,

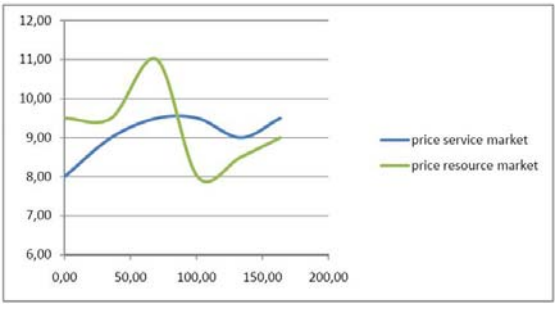


Fig. 3. Price evolution

seven BSAs and four RAs. More than 20 interactions have been fulfilled. The agents follow a very simple strategy and use their reservation price when bidding in an English Auction. If they succeed, they decrease their own reservation price, otherwise they increase it in order to be able to win the next auction.

This results could be easily compared with simulations using other negotiation protocols (for instance a Dutch auction protocol), other reputation models and other agent strategies[2].

5 Conclusion and Future Work

We have presented a simulation platform that uses agents, electronic institutions and the paradigm of the Internet-of-Services. This seems to be a promising combination when talking about service markets where reputation-based trust plays an important role. The main characteristics of an agent participating in both different environments have been extracted, and merged into one that is able to participate in a simulated environment like that one. With our simulation tool based on the Electronic Institutions, it will be possible to vary different parameters regarding policies in the Internet of Services, i.e. the interaction protocols, the reputation models or the network topology of the overall system. Our tool set can contribute to the vision of Internet of Services and service markets becoming reality as it alleviates the side effects, which occur by introducing markets to trade resources and services. Finally, changing the simulation core against a real middleware implementation will provide a proof-of-concept system as future work.

Acknowledgments

This work was supported by the EC under the FP6 programme [eRep project CIT5-028575]; the Spanish Education and Science Ministry [AEI project TIN2006-15662-C02-01, AT project CONSOLIDER CSD2007-0022, INGENIO

[2] The source code of a beta version of the platform together with this particular simulation can be downloaded at http://sourceforge.net/projects/erepsim

2010]; Proyecto Intramural de Frontera MacNorms [PIFCOO-08-00017] and the Generalitat de Catalunya [2005-SGR-00093]. Daniel Villatoro is supported by a CSIC predoctoral fellowship under JAE program.

References

1. Kesselman, C., Foster, I.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco (1998)
2. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: An open grid services architecture for distributed systems integration (2002)
3. Streitberger, W., Hudert, S., Eymann, T., Schnizler, B., Zini, F., Catalano, M.: On the simulation of grid market coordination approaches. Journal of Grid Computing; Special Issue on Grid Economics 6(3) (2008)
4. Eymann, T., König, S., Matros, R.: A framework for trust and reputation in grid environments. Journal of Grid Computing; Special Issue on Grid Economics 6(3), 225–237 (2008)
5. Akerlof, G.A.: The market for 'lemons': Quality uncertainty and the market mechanism. The Quarterly Journal of Economics 84, 488–500 (1970)
6. Pinyol, I., Sabater-Mir, J., Cuni, G.: How to talk about reputation using a common ontology: From definition to implementation. In: Proceedings of the Ninth Workshop on Trust in Agent Societies, Hawaii, USA, pp. 90–101 (2007)
7. Foster, I., Jennings, N., Kesselman, C.: Brain meets brawn: Why grid and agents need each other (2004)
8. Bell, W.H., Cameron, D.G., Millar, A.P., Capozza, L., Stockinger, K., Zini, F.: Optorsim: A grid simulator for studying dynamic data replication strategies. International Journal of High Performance Computing Applications 17(4), 403–416 (2003)
9. Buyya, R., Murshed, M.: Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency and Computation: Practice and Experience (CCPE) 14, 1175–1220 (2002)
10. Esteva, M.: Electronic Institutions: from specification to development. IIIA PhD Monography, vol. 19 (2003)
11. Brito, I., Pinyol, I., Villatoro, D., Sabater-Mir, J.: Hiherei: Human interaction within hybrid environments. In: The Eighth International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2009 (2009)
12. Esteva, M., de la Cruz, D., Sierra, C.: Islander: an electronic institutions editor. In: Proceedings of AAMAS 2002, Bologna, Italy, pp. 1045–1052 (2002)
13. Esteva, M., Rodríguez-Aguilar, J., Rosell, B., Arcos, J.: Ameli: an agent-based middleware for electronic institutions. In: Proceedings of AAMAS 2004 (2004)
14. Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)

Applying JIAC V to Real World Problems: The MAMS Case

Alexander Thiele, Thomas Konnerth, Silvan Kaiser, Jan Keiser, and Benjamin Hirsch

Technische Universität Berlin, DAI Labor

Abstract. In this paper we describe the execution platform for the MAMS service framework which provides an infrastructure for the creation, deployment and execution of service compositions created by non-IT-experts. The MAMS framework consists of an elaborate graphical service creation environment as well as the service execution platform based on intelligent agents which we describe in this paper. Our platform provides a flexible service execution environment, which utilizes agent technology to improve scalability, management and stability. Furthermore, it serves as a testing ground for enhanced features like service matching, runtime load balancing and self healing mechanisms.

1 Introduction

By now, the technologies for global provisioning of services are well established, but it is still difficult for small and medium enterprises (SME) to act as service providers. The reasons for this come from the need for technical know-how to create services and the necessary infrastructure to provide the created services. The existing service authoring tools are not intended to be used by non-technical persons and the needed hardware as well as the software to run the services must be purchased, configured and maintained, which makes the provisioning of services costly and therefore often unprofitable for SMEs.

The project MAMS (Multi Access - Modular Services) [16] addresses these issues by allowing non-technical persons to fast and easily create, deploy and manage services, according to the users needs.

With the incorporation of agent technology, we have realized an open distributed service delivery platform (ODSDP) as part of the overall MAMS framework. The ODSDP is built using the JIAC V [12] agent framework and addresses several important issues relevant for service provisioning, e.g. deployment, execution and management. It also enables adaptability to dynamic environments.

In this paper, we first describe the project context from which we derive requirements for our service delivery platform. In Section 3) we introduce the agent-based ODSDP for deployment, execution and management of services which is followed by a report of experiences with the framework. In the next section we give an overview of previous and related work and finally, a short conclusion and a precise outlook of future work is given.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 268–277, 2009.

2 The MAMS Project

The development of new services for telecommunication applications and other IT systems is one of the most important vehicles of innovation for telecommunication and other service providers. Competition and ever growing expectations of users require a faster service development and the means to deliver those services promptly. Thus, the acceleration of the service development process requires the development of new concepts and tools. The MAMS project [4] develops tools and agent-based approaches as answers to these problems.

It is not the focus of this paper to go deeply into the details of the MAMS framework as we want to describe the multi-agent based service delivery and execution platform. Instead, we will show the process from service creation to execution here and derive requirements that motivate our work.

In MAMS new services are created by combining several basic services into a service composition. The combination is data flow driven where output ports of a basic service are connected to input ports of another service. The user, a non-IT-expert, is supported by a visual service creation environment where new services are created easily by drag 'n drop and connecting matching ports.

Following the creation process, the service compositions are deployed into a runtime environment and may afterwards be started and used via a webinterface. This leads to certain requirements for the runtime environment. First of all, it is obviously necessary to provide a deployment method for service compositions at runtime. These service compositions have to be executed at some point, which requires the ability to find and invoke referenced basic services dynamically. One step further, we also want to provide additional reliability and fault tolerance with the ability to find alternatives for these references services, be it because a basic service is broken, or because it cannot be found any more.

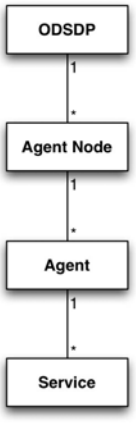


Fig. 1. ODSDP structure

Additionally, as these service compositions are generated by non-experts who are not familiar with the execution environment, we need to make sure that the execution happens in a controlled environment, because we do not want flawed compositions to affect the whole platform. In order to ensure scalability and responsiveness in the case of large numbers of service compositions, we also need mechanisms to balance and control the resources consumed by the execution, which we address with appropriate load balancing mechanisms.

Finally, our runtime environment has to provide support for the development and deployment of basic services. We need an embedded user management, which provides methods for authentification and authorization. Additionally, the functionalities for basic services are often already available in the form of web services or other libraries, so our runtime environment needs to be able to access other technologies in order to utilize these functionalities.

3 Service Execution Framework

The ODSDP is the agent-based service execution environment of the MAMS framework. As shown in Figure 1 the ODSDP consists of agent nodes, i.e. physical software environments bound to actual machines, where agents are deployed which provide the services. All agent nodes are connected by a Java Message Service(JMS) based message broker. This allows an organisational as well as a physical distribution of the services. Additional to the message broker and a comprehensive Java Management Extensions(JMX) based management interface, optional agent node components may provide infrastructure functionality, e.g. a distributed service directory connected by the message broker for announcing and searching of services in a peer-to-peer manner without a centralized database. A caching mechanism can be activated to speed up the global search. Both kinds of services (basic services and composed end-user services) are provided by agents, which can be deployed on agent nodes at runtime by using the management interface as well. Advantage of this approach is that the services can be transparently distributed over different nodes, and that they have their own knowledge and an own thread of control instead of being dependent on a shared engine.

Agent Technology in MAMS

The enabling technology for integration and inter operation of the different components of the MAMS Framework is a service execution environment based on a multi agent system. While some of its concepts and features are presented in the next sections, here we argue that using a multi agent system for service execution environment has several advantages.

First, when used as an organizational and functionally closed entity, agents provide straight forward means to map functionalities such as basic services or service compositions to logical entities such as agents. Thus, a clear separation of both entities is enforced by an multi-agent-system. Based on this mapping, it is now possible to provide each service stakeholder (e.g. platform provider) with a set of introspective and - if applicable - management features that again are clearly separated. As a consequence, the agent management infrastructure can be extended with more features like accounting without the need to rearrange or redesign the service execution environment. Another important aspect of the organizational agent-to-entity mapping is the fact that the distributed system is highly scalable. In general, the only bottleneck are available platform resources, especially network bandwidth, since CPU power and storage capacity can easily be added. While the above mentioned advantages are aspects of the system design, there are other more functional aspects of agent technology that can be usefully applied in the context of a service execution environment. Agent technology and especially multi agent systems are well suitable for creating systems that incorporate adaptive behaviour [8,15,5,7]. We have have designed adaptive agents and mechanisms that deal with dynamic load balancing mechanisms and self healing. We will refer to these scenarios in the next sections and in section future work.

Agent Architecture

A MAMS agent (as shown in Figure 2) is based on a modular and manageable component architecture, which contains two mandatory components: a memory for the storage of agent-internal knowledge and an execution cycle to enable proactive behaviour and scheduling of all actions. Optional application-independent components allow amongst others the communication with other agents by using the message broker of the agent node, the access to the agent node's service directory by using the communicationcomponent, the semantic matching of services by using the access to the service directory, the processing of rules by using the service execution, and the execution of composed services by using the service matcher, communication and service adaptors.

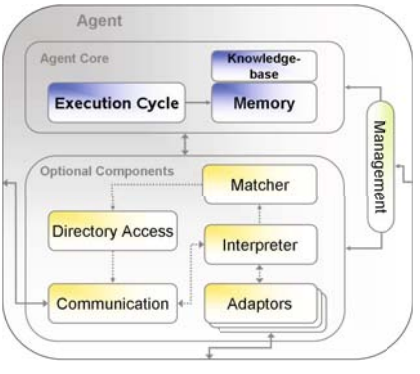


Fig. 2. A MAMS agent

These adaptors either directly provide basic services or generically integrate services based on other technologies, e.g. web services, UPnP or FIPA. The interaction between all components inside an agent (see dotted arrows) takes place via the memory according to the blackboard metaphor, i.e. each component has access to the memory and is triggered by the execution cycle. Security mechanisms such as Single Sign-On(SSO) based authentication and authorization ensure that the management interface and the services can only be used by these groups of users which are specified by the responsible service provider or platform provider.

Service Compositions with DFL

The service composition language itself, is based on the JIAC Agent Description Language (JADL) [13] and it also unites ontology based data structures and service-oriented programming. The declarative part of the DFL is completely based on OWL [17], including OWL-S [3] for the service descriptions. This allows us to resort to existing reasoning and service matching implementations and concentrate on the integration into our agents.

The new feature of the scripting language however is the notion of *abstract service invocation*. Each invocation of a basic service within a service composition (i.e. a DFL script) is represented by the invocation of a service template. This service template is basically an OWL-S description that may be incomplete. This approach has the advantage that the same mechanism can be used for precise service invocation (if the parts of the template that are provided identify the service unambiguously) or for a goal driven approach. In the latter case, the template is regarded as a goal and the agent tries to find and execute a service that matches the template. By using the preconditions and effects that

are given in an OWL-S description, we can improve the matching process, verify the results, and even apply planning from first principles if appropriate.

Another advantage of using an interpreted scripting language is, that we can fully control the execution of a service composition. This means that we can stop the execution and resume it later, at any point of the composition. The fact that each service composition is deployed in its own agent greatly enhances the possibilities for monitoring and management of the ODSDP.

Load Balancing

As mentioned in section above agent technology offers many possibilities to apply adaptive behaviour. We have implemented an adaptive load balancing mechanism that incorporates a two stage load balancing process - start time and runtime load balancing - based on resource evaluation. This approach is needed in order to react to the deployment of new services at runtime by controlling the distribution of services among the different Nodes. Furthermore stability of service execution is increased by preventing and handling high system load.

Load balancing ensures that the platform optimises usage of available resources. Services that use a lot of resources should run on hosts that have few to none other services running in parallel or that can provide the resources needed. Load balancing on the ODSDP therefore aims distributing services in a running system. This mechanism focuses on improved resource usage (CPU utilisation, memory, network and hard disk storage capacities) in a best effort approach.

Start time load balancing is a common strategy applied by most cluster and grid systems and relies on a scheduler that assigns new jobs to specific hosts according to current load parameters. When new agents are deployed, a specific component checks the current load of the platform by using the Load Reporter services. Based upon the available load data and several coordination requirements, it is then decided at which node the new agent is to be deployed.

As service usage and thus system load in a multi agent system is highly dynamic, forecasting load distribution at start time is very inaccurate. Thus, a runtime load balancing mechanism is applied to react to local load peaks. Runtime load balancing dynamically relocates services while they are running and thus can react to the development of system load on the active hosts. This approach is primarily used for long running jobs and less useful for jobs that have short lifetimes [11,25]. At runtime, a specific component periodically checks the current load of the local node. If high system load is detected, it tries to migrate local agents to other nodes taking into account several coordination considerations to prevent useless migrations and oscillations. However, transferring an agent is a costly process, thus transfers take place only if the load of the target node is less than the load of the source node, with a significant load difference in order to prevent migrations at minimal load differences. Other constraints need also be evaluated, like resource availability.

Self Healing

We consider Self Healing as being a phrase referring to the automated detection of specific types of system errors in conjunction with an automatically generated

countermeasure that resolves the error and lets the system perform its intended operations.

In the context of service execution, possible errors that might occur are service failure, failure of resources e.g. connection to legacy systems or connections to other nodes. We focus on errors that directly prevent service usage and use redundant strategies as well as a straight forward mapping of errors to countermeasure strategies to resolve the error. In a first scenario we have applied service matching technology to circumvent failure of basic services.

In order to invoke a basic service while executing a service composition the DFL interpreter needs to search the service directory in order to find out where this service is located and who provides the service. The service directory is a good place to detect most types of service failures. The directory sends a special keep-alive-message to each registered service to check whether the service does respond. The service has to answer within a certain time. This way it can be detected if the service is physically accessible and if it can be invoked. In case a service does not respond the service directory removes the service from its list of known services. Search requests for this service are then prompted with an error. The DFL interpreter receives the error and reacts in a defined manner. It tries to substitute the failed service by invoking a service matching based on precondition and effects. If a substitute service can be found it is invoked and execution of the service composition can be continued. By applying a redundant strategy we can increase chances that a substitute service might be found. A simple approach is to deploy each service at least twice such that each service runs on a different node.

4 Experiences and Results

During the MAMS project, we had multiple test scenarios for the verification of our approach. We cooperated with different partners including specialists in the areas of health care and telecommunications as well as small and medium enterprises to create realistic scenarios for the testing the framework and our multi agent platform.

One of these testing scenarios was the creation of a diary of vitals signs for rehab patients. This diary is composed of multiple service compositions that allow both, the patient and the attending physician to review and analyse the entries of the diary. The patient can access his diary via the web, make new entries, and have both, himself and his physician be notified via different messaging technologies, if any uncommon or critical measurements occur.

The service compositions that constitute this diary typically consist of calls to the UI-Service, one or more calls to the persistence services and sometimes a messaging service in the end. Additionally, physicians and patients may use service compositions that enable a video conference via the IMS-based services.

Scenarios like this were used to evaluate the complete processes defined in MAMS for the creation of services by non-IT-experts, including service composition, deployment and service execution.

Our testing scenarios showed, that we are able to quickly deploy both, basic services and service compositions, on a running platform. We were able to ensure stability and scalability of all components due to the strict separation provided by the agent metaphor. This proved to be especially valuable when testing new basic services, as errors in those services did not affect the whole platform.

Regarding the agent platform, it showed that it ran stable even for multiple months. As most basic services were developed as needed in an iterative manner, we had of course regularly agents that would crash or otherwise become inactive. However, this never affected the platform or the other agents. So when one basic service crashed, the platform and the other agents would continue to work and only service compositions that used the specific defective agent would be affected. We were able to have our platform available at almost any time, which we regard as very good performance for a prototype platform.

5 Related and Previous Work

There are different fields where related work is situated. First of all there are the commercial service delivery platform from vendors such as BEA, IBM, Microsoft, or HP. Common to them is that while they provide powerful delivery platforms, they generally do neither address the convergence between IT and telecommunications, nor do they deal with semantic service descriptions and their execution [20].

While easy service creation is not the focus of this work, we want to mention a number of European projects there are working in this area. OPUCE [19] provides a graphical tool based on a modelling language similar to UML for easy composition of services by non-professional service providers. SPICE [24] in addition considers semantical information to allow a more automatically composition of their rule-based services. LOMS [14] uses a template approach to support really non-technical service providers by integrating an additional service operator role. There has been an increasing focus within the agent community on the interaction of agents and services, and a number of workshops and textbooks (e.g. [23]) deal with the topic.

Many researchers looked at the relevance of webservices agents, for example [8,5,26]). The integration of webservices into agents has been looked at by e.g. Greenwood and Calisti [10] who developed a webservice gateway mediates between FIPA agents and webservices. Zinnikus et al. [27] follow a similar approach where the agent's actions are exposed as webservices. Using these kind of approaches, a certain flexibility within service orchestrations can be achieved.

Others deal with the integration of workflows and agents, which is closer to the aims of the MAMS project. Already during the mid90ties, the ADEPT project used agents that provided services and organised themselves to achieve business goals ([2,18]). The focus here was however on the use of workflow and not so much on the technical issues like deployment.

Singh et al. used agents to manage workflows [22]. The WADE system [6] extends JADE to allow agents to orchestrate methods to workflows. The system

can however not be used by non-experts. As the engine compiles the workflows down to Java byte code, the tight execution control and its associated features are not available.

Most of these approaches come either from a rather theoretical point of view and do not address the needs of a modern IT-infrastructure or they do not cover the semantical layers that are necessary for using ontologies and service matching. At the DAI Labor, service oriented agents have been used for some time in the JIAC IV framework(see e.g. [1,9,21]). While JIAC IV had powerful features such as planning from first principles and integrated ontologies [13], it was not up to the requirements of the MAMS system.

As discussed above, the existing platforms were not able to provide an environment for service execution that could support stable and secure service deployment and provision coupled with a semantic approach that allowed for abstract services. This was our main motivation for creating an agent framework that merges service technologies with semantical descriptions and reasoning.

6 Conclusion

In this paper we have demonstrated the usefulness of applying agent technology in the context of service execution. We have created an agent-based service delivery platform - the ODSDP - which incorporates important features such as scalability and manageability. We have shown that these features can be efficiently integrated into a service execution environment with the help of agent-oriented software design. By assigning each new service composition to a single execution agent, runtime deployment and control of service execution could be achieved. We have successfully applied adaptive service execution strategies by including load balancing and semantic service matching mechanisms. We conclude that multi agent technology is of great benefit in the design and implementation of a service execution environment as shown in this paper.

Future Work

Based upon the components described in Section 3, two load balancing strategies have been designed. There are still open issues though. The start time load balancing process contains no queuing mechanism and thus does not prevent a general overload of the whole platform. At runtime the selection of the local agent to be migrated can be used to further improve load balancing. Instead of picking a random agent, strategies for selecting the most appropriate candidate (e.g. with respect to resource usage) would further improve runtime load balancing and are topic of future work.

The service matching process could be further enhanced but finding a service chain that could substitute a failed service. Because service matching and planning chains of services can be time consuming these processes would need to run in background, e.g. while the platform has plenty of resources left, not just in case of failure. The service directory could then trigger service matching and provide results on the fly in case of service failure.

While the notion of "self" implies detection and reaction by the platform itself, it is still a great challenge to actually learn to detect and react to errors which the system has not been told to recognize and handle. Especially finding the "right" actions in production systems proves to be difficult because it is not clear how these actions would effect the overall stability and functioning of the platform, since the system might perform countermeasures that have unwanted side effects. We think that nature inspired approaches or self organization combined with traditional learning mechanisms (e.g. neural networks) might provide ways in resolving these issues.

Acknowledgements. We would like to thank Axel Hessler and the Competence Centre Agent Core Technologies for their support.

This work has been sponsored by the Federal Ministry of Education and Research. (Project funding reference number 01BS0813).

References

1. Albayrak, S., Wieczorek, D.: JIAC - an open and scalable agent architecture for telecommunication applications. In: Albayrak, S. (ed.) Intelligent Agents in Telecommunications Applications - Basics, Tools, Languages and Applications. IOS Press, Amsterdam (1998)
2. Alty, J.A., Griffiths, D., Jennings, N.R., et al.: Adept - advanced decision environment for process tasks: Overview & architecture. In: Proc. BCS Expert Systems 96 Conference, pp. 5–23 (1994)
3. Barstow, A., Hendler, J., Skall, M., et al.: OWL-S: Semantic Markup for Web Services (2004), http://www.w3.org/Submission/OWL-S/
4. Freese, B., Stein, H., Magedanz, T., Dutkowski, S.: Multi-access modular-services framework – supporting smes with an innovative service creation toolkit based on integrated sdp/ims infrastructure. In: 11th Int. Conf. on Intelligence in Service Delivery Networks (2007)
5. Bozzo, L., Mascardi, V., Ancona, D., Busetta, P.: CooWS: Adaptive BDI agents meet service-oriented programming. In: Isaias, P., Nunes, M.B. (eds.) Proceedings of the IADIS International Conference WWW/Internet 2005, vol. 2, pp. 205–209. IADIS Press (2005)
6. Caire, G., Gotta, D., Banzi, M.: WADE: A software platform to develop mission critical applications exploiting agents and workflows. In: Berger, M., Burg, B., Nishiyama, S. (eds.) Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems, May 2008, pp. 29–36 (2008)
7. Casella, G., Mascardi, V.: From AUML to WS-BPEL. Technical Report DISI-TR-06-01, Dipartimento di Informatica e Scienze dell'Informatione, Università di Genova (2006)
8. Dickinson, I., Wooldridge, M.: Agents are not (just) web services: Considering BDI agents and web services. In: Proceedings of the 2005 Workshop on Service-Oriented Computing and Agent-Based Engineering (2005)
9. Fricke, S., Bsufka, K., Keiser, J., et al.: Agent-based telematic services and telecom applications. Commun. ACM 44(4), 43–48 (2001)
10. Greenwood, D., Calisti, M.: Engineering web service - agent integration. In: IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1918–1925 (2004)

11. Harchol-Balter, M., Downey, A.B.: Exploiting process lifetime distributions for dynamic load balancing. ACM Trans. Comput. Syst. 15(3), 253–285 (1997)
12. Hirsch, B., Konnerth, T., Heßler, A.: Merging agents and services — the JIAC agent platform. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Springer, Heidelberg (2009)
13. Konnerth, T., Hirsch, B., Albayrak, S.: JADL — an agent description language for smart agents. In: Baldoni, M., Endriss, U. (eds.) DALT 2006. LNCS (LNAI), vol. 4327, pp. 141–155. Springer, Heidelberg (2006)
14. LOMS. Local mobile services, http://www.loms-itea.org
15. Luck, M., McBurney, P., Shehory, O., Willmott, S.: Agent Technology: Computing as Interaction (A Roadmap for Agent Based Computing). In: AgentLink (2005)
16. MAMS. MAMS multi access - modular services, http://www.mams-platform.de/
17. Martin, D., Hodgson, R., Horrocks, I., Yendluri, P.: OWL 1.1 web ontology language (2006), http://www.w3.org/Submission/2006/10/
18. O'Brien, P.D., Wiegand, W.: Agent based process management: Applying intelligent agents to workflow. Knowledge Engineering Review 13(2), 1–14 (1998)
19. OPUCE. Open platform for user-centric service creation and execution, http://www.opuce.tid.es
20. Preuveneers, D., Pauty, J., Van Landuyt, D., et al.: Comparative evaluation of converged service-oriented architectures. In: 21st Int. Conf. on Advanced Information Networking and Applications, vol. 1, pp. 989–994 (2007)
21. Sesseler, R., Albayrak, S.: JIAC IV - an open, scalable agent architecture for telecommunications applications. In: Proc. of the 1st int. NAISO Congress on Autonomous Intelligent Systems. ICSC Interdisciplinary Research (2002)
22. Singh, M.P., Huhns, M.N.: Multiagent systems for workflow. Int. Journal of Intelligent Systems in Accounting, Finance and Management 8, 105–117 (1999)
23. Singh, M.P., Huhns, M.N.: Service-Oriented Computing. Wiley, Chichester (2005)
24. SPICE. Service platform for innovative communication environment, http://www.ist-spice.org
25. Stender, J., Kaiser, S., Albayrak, S.: Mobility-based runtime load balancing in multi-agent systems. In: Proceedings of the 17th Software engineering and knowledge engineering conference (SEKE), KSI, 3420 Main Street, Skokie, IL 60076, USA, p. 688. Knowledge Systems Institute (2006)
26. Walton, C.: Uniting agents and web services. In: Agentlink News. AgentLink, vol. 18, pp. 26–28 (2005)
27. Zinnikus, I., Hahn, C., Fischer, K.: Model-driven, agent-based approach for the integration of services into a collaborative business process. In: Pagham, Parkes, Müller, Parsons (eds.) Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems, pp. 241–248 (2008)

Agent-Based Semantic Search at motoso.de

Nils Weber[1], Lars Braubach[2], Alexander Pokahr[2], and Winfried Lamersdorf[2]

[1] motoso.de GmbH & Co. KG
nilsweber@gmx.de
[2] Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
{braubach,pokahr,lamersdorf}@informatik.uni-hamburg.de

Abstract. Searching for information in large rather unstructured real-world data sets is a difficult task, because the user expects immediate responses as well as high-quality search results. Today, existing search engines, like Google, apply a keyword-based search, which is handled by indexed-based lookup and subsequent ranking algorithms. This kind of search is able to deliver many search results in a short time, but fails to guarantee that only relevant data is presented. The main reason for the low search precision is the lack of understanding of the system for the original user intention of the search. In the system presented in this paper, the search problem is tackled within a closed domain, which allows semantic technologies to be used. Concretely, a multi-agent system architecture is presented, which is capable of interpreting a key-words based search for the car component domain. Based on domain specific ontologies the search is analyzed and directed towards the interpreted intentions. Consequently, the search precision is increased leading to a substantial improvement of the user search experience. The system is currently in beta state and it is planned to roll out the functionality in near future at the car component online market-place motoso.de.

1 Introduction

Searching for information is a difficult task and is considered to be an important skill for internet users. Despite this difficulty, currently well known search engines like Google are mainly based on keyword-oriented search requests. A search process often involves several rounds in which the user has to iteratively refine the query according to the preliminary gathered results. One reason for this way of searching is that the request style is easy to employ for the users. Nevertheless, the user intentions cannot be understood well, because only the occurrence of certain keywords decides about the result relevance and its inclusion in the result set. Understanding user intentions in general is very hard but becomes easier when the search topic is constrained in beforehand e.g. by considering a closed domain only. In this case semantic technologies, like ontologies and corresponding reasoning mechanisms, can be applied to analyze and comprehend the query. Such an approach has the potential to combine the ease of use of a keyword-oriented search request with the semantic expressiveness of a metadata based query.

L. Braubach et al. (Eds.): MATES 2009, LNAI 5774, pp. 278–287, 2009.

In this article, semantic search techniques are applied to the car domain at the German Internet marketplace motoso.de [11]. The platform is mainly specialized in new and used replacement and tuning components, tires and wheels as well as garage services for all types of vehicles. Currently, the total number of components is about 5.8 Million, whereby 85% is located in the passenger car category. In addition, roundabout 400.000 complete automobiles are registered in the database. The adverts are primarily placed by commercial customers and can contain a lot of detail information that is possibly of interest for potential buyers. In order to manage the adverts they are grouped into a tree-like structure with main and subcategories (e.g. 16 main categories for components with circa 700 branches of varying depth), which also exhibit links to the car types as well as their model and variant refinements. In addition to the base information containing a short and long description, price, state, origin, etc., a lot of category specific attributes with different types and allowed values can be specified. This allows a very precise (semi-structured) description, which should enable potential customers to find adequate offers. In this context, one serious problem is that descriptions are entered by different service providers in very different ways leading to variable data quality. This variable data quality causes non optimal search results, because the search is based on full text indices and might not take into account relevant but poorly described adverts.

In the next section the background and related works concerning semantic search is shortly summarized. Thereafter, in Section 3 the system architecture of the agent-based semantic search engine is presented. Its realization within motoso.de is described in Section 4 and the paper closes with a summary and an outlook on planned future work.

2 Analysis of the Search Problem

The motoso.de portal can be seen as representative for a wide range of internet- or intranet-based systems, i.e. the class of systems, which provide a search function as the main entry point to access its contents. These kinds of systems share a number of common properties:

- a common theme or domain for the whole portal,
- huge data sets,
- large diversities of data contents,
- heterogeneity with respect to data quality and data completeness (e.g. due to heterogeneous data sources),
- multiple search alternatives including a 'flat' keyword-based quick search as well as extended search and/or browsing capabilities based on domain specific metadata and categorizations.

The assets of such a kind of portal are the stored entries, where entries can be offered goods or services in marketplace like eBay (or motoso.de), but also e.g., publications in a scientific research portal. The main purpose of the portal's

web interface is to provide an interested user access to those (and only those) entries that match its current interest. The different search alternatives provide the means for the user to express this interest to the system. As an example, two search alternatives are described in the following and discussed with respect to their advantages and disadvantages.

Quick Search. A single search phrase can be entered that (by default) leads to a keyword search matching those entries where all words appear in some part of the entry.

Advanced (or extended) Search. The user is offered detailed control over which parts of the entries data should be searched (e.g. title, text, specific attributes). Moreover, queries a not entered as 'flat' text but follow domain specific representations (e.g. two number fields for entering a price range, a drop down list or a set of check boxes for choosing among predefined categories).

Quick search is the kind of search that most search engines like Google offer and that is therefore well known by virtually every internet user. Among the biggest advantages of the quick search is its intuitivity and ease of use. The disadvantage is the generally poor result quality of purely keyword-based search [1,10]. Therefore, most search engines improve the *perceived* result quality by applying stemming, and removing stop words from the search phrase and by sorting the results using complex ranking algorithms. Nevertheless, this approach can not incorporate domain knowledge for interpreting search requests.

The advanced search approach requires that knowledge about the domain of the portal is explicitly incorporated into the system (for providing differentiated search options), but also assumes that the user is knowledgeable in the domain itself (to make use of the advanced search options). The disadvantage of this approach is therefore the complicated query formulation that might deter inexperienced users. On the other hand, experienced users benefit from the ability of formulating very precise queries with respect to the relevant properties of the considered domain.

The two described search alternatives form extremes with respect to the trade-off between usability and result quality. Recently, semantic approaches have been introduced that try to combine both advantages. The general idea of these approaches is to offer a simple free form query, but use semantic technologies for interpreting the query and thus improving search results. An overview of several of these approaches is given in the rest of this section. The overview is coarsely divided into semantics for broad-based search engines vs. semantically enriched vertical, i.e. domain-specific, engines.

Introducing semantics is typically much more ambitious for broad-based search engines, because they cannot easily draw initial implications from the query. An elegant solution for this problem are semantic web search engines that are capable of performing an attribute-based retrieval process on typical semantic web resources like ontology A-boxes and RDF resources. These engines can directly make use of semantic technologies like ontological reasoners but currently can

operate on a small database only. Regrettably, they are subject to the basic chicken and egg problem of the semantic web, i.e. people don't see the reason to be one of the early adopters for meta tagging their sites because the merits of this overhead remain unclear as long as no critical mass has been achieved [5]. Examples of this category are the SHOE[1] and Swoogle[2] [4] search engines. Other approaches like Powerset[3] and Cognition[4] (e.g. compared in [7]) try to semantically interpret the query using natural language processing (NLP) techniques. Basically, these approaches translate the user query into a canonical query over the database of indexed documents [6]. Some NLP search approaches also classify the interpreted query according to several predefined areas of expertise. In a second step these approaches can then use domain-specific for interpreting the categorized query similar to vertical search engines (see next paragraph). An example of this approach is the Hakia search portal[5], which supports queries in multiple languages and covers already a diversity of different domains.

Vertical search engines are developed for specific domains and include domain knowledge for retrieving optimized search results. Classical vertical search engines operate similar to standard search engines and use indices for looking up search results. These engines have clear advantages in the crawling and indexing process, because the spiders only have to be sensitive for web pages that contain keywords from the considered domain and the index is also constrained by the domain vocabulary [12]. The quality of search can be further enhanced when vertical search engines are enriched with semantic technologies. An ontology for the target domain can be developed by experts from the specific area. The ontology can then be used to understand terms and possibly also combinations of terms in the query leading to a (more or less vague) interpretation of the user intention. As described above, for this interpretation, sophisticated NLP algorithms can be applied, though, at least some indication exist [3] that a vague but simple ontology-based interpretation of a query can for some application cases be more appropriate than a thorough NLP analysis. One example is the vertical semantic search engine UpTake[6] for holiday planning. It tries to extract the general intention of the trip planning such as "family holidays" versus a "romantic trip". On basis of this broad categorization the search is directed to the right data by knowing ontologically, what makes up the interpreted type of trip. This information can then be used to rank the results accordingly.

The system described in this paper uses ontology-based reasoning for improving the search in a domain-specific portal. It therefore follows a similar approach like UpTake. The main reasons for this choice are its relative simplicity and effectiveness compared to other (e.g. NLP) approaches.

[1] http://www.cs.umd.edu/projects/plus/SHOE/search/
[2] http://swoogle.umbc.edu/
[3] http://www.powerset.com/
[4] http://www.cognition.com/
[5] http://www.hakia.com/
[6] http://www.uptake.com/

3 Semantic Search System Architecture

In this section the general architecture of the semantic search system and its implementation will be introduced. Its main purpose is to transform an incoming quick search query to an advanced query with explicit structure via interpretation of the formerly unstructured search phrase. In addition, the system shall also be capable of improving its behavior constantly by incorporating user feedback on the search quality.

3.1 System Design

The main architecture of the ASQP (agent-based semantic query processing) system is depicted as Prometheus [8] system overview diagram in Fig. 1. An incoming quick search request is directed towards the *monitor* component that is responsible for its complete processing and therefore can be considered as the control unit of the system. In a first step the control unit decomposes the request into a list of single words and word chunks up to a predefined length and distributes it together with the original request to all active *parsers*. Each parser analyzes the request according to its domain knowledge, whereby the system can provide an arbitrary number of differently specialized parsers. Hence, a parser tries to identify the semantics of the single tokens and small token chunks of the request by operating on a specific ontology or taxonomy and sends its interpretation back to the monitor. The monitor collects the answers and in case the last answer arrived or a defined timeout occurs preevaluates the results. If there are conflicting interpretations of words or chunks from different parsers the results are forwarded to a *mediator*. The mediator has access to all

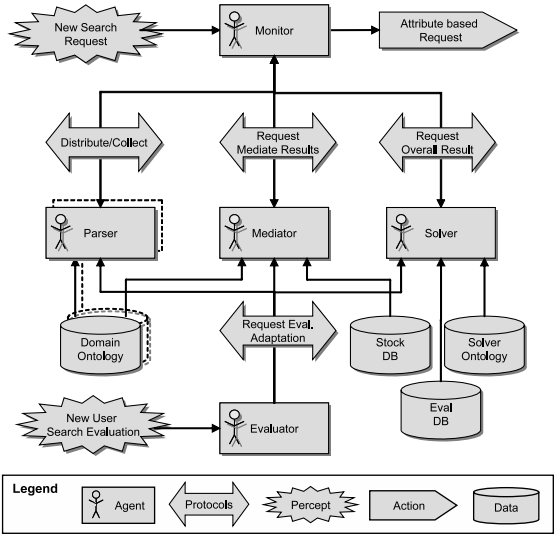


Fig. 1. ASQP system architecture

ontologies, previous evaluations and probabilistic observation data and decides for each conflicting word or token which decision to follow. The arbitrated results are then forwarded to a *solver,* which has the purpose to generate the final solution. Therefore, it mainly resorts to an ontology of search pattern behavior, in which search intentions are stored for groups of semantic components. The search intention together with the semantically analyzed request is then handed over to the monitor, which redirects the request as attribute-based search to other established components outside of the ASQP system.

The incorporation of user feedback in the system is realized via a dedicated *evaluator* component. This component is fed with user evaluation data concerning single requests and calculates evaluations for the parsers, mediator and solver. These evaluations are stored in a specific evaluation database, and allow estimating the system's standard of performance. Furthermore, the specific evaluations are also forwarded to the corresponding components, which in this way get a chance to adapt their behavior accordingly.

3.2 System Implementation

The ASQP architecture has been implemented as multi-agent system using Jadex 0.96 [2]. Each of the roles, as defined in the design, has been mapped to a separate BDI agent type. The agents operate on the RDF and OWL ontologies using the semantic web framework Jena2[7]. In order to meet the performance criteria for a real-time search engine, most ontologies have been indexed using LARQ (Lucene + A SPARQL Processor for Jena). This allows a quick full text search on the ontologies with LARQ-extended SPARQL [13] queries. To reduce the number of matches in beforehand of the processing, additionally a Lucene-score can be used. Reasoning capabilities are currently only used in the solver agent. This agent instantiates an inference ontology consisting of the parser results and applies (currently quite simple) inference rules to deduce the user intention. Finally, the integration of the system with the existing search infrastructure and with the user interface had to be considered. The connection to the existing system infrastructure is minimal and is restricted to a search call on the traditional attribute-based search service. On the other hand, the user interacts via the browser with the system the Jadex webbridge framework [9] could used to simplify the delegation of user requests via Java servlets to the agent tier and back. Finally, the user interface of the portal itself was changed slightly. The user now has the possibility to explicitly turn on/off the semantic search facility. In addition, a new feedback dialog is generated, in which a user can state if the extracted intention was right or not.

4 Realization within motoso.de

To incorporate the ASQP system into the motoso.de portal[8], two distinct tasks had to be performed. For the system to provide meaningful results with respect

[7] `http://jena.sourceforge.net`

[8] A test version of the motoso.de semantic search is available at: `http://semanticmotoso.mine.nu/`

to the automotive domain, specific ontologies and inference rules had to be provided. Additionally, different modes of operation were implemented, providing different strategies, how the system interacts with a user.

4.1 Domain-Specific Ontologies and Inference Rules

Domain knowledge for the motoso.de portal is reflected in three separate ontologies, which are used by the respective parser agents (cf. 3.1). The purpose of each of the parser agents is to assign meaning to single keywords or phrases. Each parser only considers a specific ontology and therefore only assigns meaning to those keywords that match some concept of this ontology. For the interpretation of the ontological concepts, all domain-specific ontologies refer to a common general-purpose ontology containing basic concepts and properties such as 'isPartOf' and 'equivalent'. The domain-specific ontologies, described below, all contain type as well as instance data.

Car Brands and Model Series Ontology. On the type level, this ontology introduces broad concepts like 'Manufacturer' and 'VehicleType', but also very detailed conceptual structures for describing properties of model variants, etc. The instance level contains data about a wide range of car types and is taken from the corresponding German federal authority (Kraftfahrtbundesamt).

PARTS Ontology. PARTS is an acronym for (replacement) parts, accessories, rims/tires, tuning parts, and services. The PARTS ontology therefore contains type and instance level data about specific parts or services. This ontology has been created by domain experts in a manual process by analyzing the existing offers. Special care has been taken to extract all the synonyms and abbreviations used for describing the same concept (e.g. 'window lifter' vs. 'window winder'), as these are important for improving the search result quality.

Geographic Locations Ontology. This ontology contains information about German cities, districts, states, and regions as well as German postal codes.

To exemplify the usage of the domain ontologies, consider the search queries 'audi 100' and 'audi 22527'. Based on the information in the car brands ontology, the first query can be matched as a complete phrase to a specific car type (Audi 100). For the second query, no match for the phrase exists, but the separate keywords can be matched, i.e. 'audi' represents a manufacturer (car brands ontology) and '22527' is a postal code belonging to a district of Hamburg (geographic ontology).

As described in Section 3.1, the combined (and possibly conflicting) meanings provided by the three parsers are collected and interpreted by the solver agent. The solver agent uses domain specific reasoning rules that operate on the results from the parsers. The reasoning rules are responsible for assessing the query intention (e.g. is the search geared towards parts or a complete vehicle). The rules are manually derived by experts and represent their experience, how unstructured queries should be matched to the domain-specific attributes. In addition, the solver can use evaluation data of previous requests to resolve conflicts due to incompatible semantic interpretations from the different parsers.

4.2 Modes of Operation

The ASQP system takes a user query, which is then augmented with semantic annotations. The system itself makes no assumptions, how these annotations should be reflected in a portals web presence. In the prototype developed at motoso.de, the user can choose from three different options of using the ASQP system:

Show. In this mode, the semantic interpretation of the users search phrase is presented to the user without altering the search itself. This mode is helpful for inexperienced users to learn how their queries would be interpreted. Moreover, this mode is used during the evaluation and training phase of the system (see below).

Suggest. Based on the semantic interpretation of the search phrase, this mode will offer alternatives these (e.g. synonymous or related concepts) in addition to the originally identified search terms.

Do. This mode transforms the original user query into a new attribute-based search, that reflects the predicted user intention.

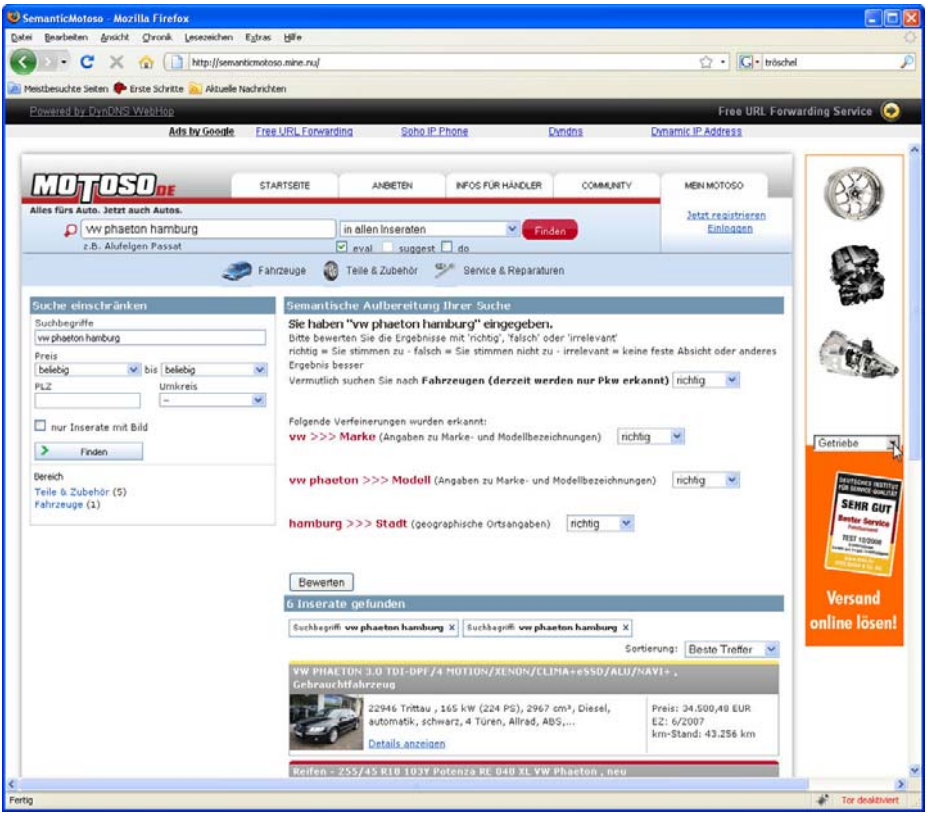


Fig. 2. Screenshot of the system in 'eval' mode

Ideally, the system would work only in 'do' mode, once it is productive, i.e. constantly providing an improved search result quality without bothering the user about the complex semantic internals that operate behind the scenes. Yet, as the dynamic content of the portal represents a moving target and the ontologies and reasoning rules are hand-crafted, it needs a certain amount of training for the system to produce stable results. Therefore a fourth mode ('eval') is implemented, that makes use of the evaluation feature of ASQP. In 'eval' mode, the semantic interpretation of an issued query is presented to the user, who can then rate each of the recognized intentions and concepts as being (in)correctly identified (see Fig. 2). The evaluation result is fed back into the ASQP system, which stores all evaluations to improve future query analysis (cf. Section 3.2).

5 Summary and Outlook

In this paper a general architecture and a concrete, domain-specific implementation of an agent-based semantic search system are presented. It is argued that semantically enriched search technology can fill a gap between an easy to use but limited 'quick search' and a powerful but complicated attribute-based 'advanced search'. An overview of existing approaches is given and related to the semantic approach used in this paper.

The proposed ASQP (agent-based semantic query processing) architecture combines semantic technology with a multi-agent system collaborative problem solving approach. The architecture is flexible and extensible in the sense that arbitrary agents, which are knowledgeable in a specific area, can easily be added and removed at any time. The different agents are thus able to interpret different portions of a user query and ultimately allow deriving user intentions from the query text. The architecture was developed in the context of a concrete application domain (the motoso.de portal) and also realized in this context. Different ontologies relevant for the car domain were devised and provided to separate agents. The results of the semantic query interpretation can be used in the portal to provide suggestions to users or directly transform the search request in accordance to the assumed user intention.

Currently, the system is running as a prototype and is used for internal evaluations of the technology at motoso.de. The evaluation is also performed for training of the system with respect to the domain in order to improve the overall query interpretation accuracy. On a technical level, the system runs stable with average response times below one second. If the ongoing evaluation ultimately confirms the initially promising results it is planned to integrate the system into the public motoso.de search interface.

References

1. Blair, D., Maron, M.: An evaluation of retrieval effectiveness for a full-text document-retrieval system. Communications of the ACM 28(3), 289–299 (1985)
2. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A BDI Agent System Combining Middleware and Reasoning. In: Unland, R., Calisti, M., Klusch, M. (eds.) Software Agent-Based Applications, Platforms and Development Kits, pp. 143–168. Birkhäuser, Basel (2005)

3. Catone, J.: Semantic travel search engine uptake launches. ReadWriteWeb (2008), http://www.readwriteweb.com/archives/semantic_travel_search_uptake.php
4. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: Grossman, D., Gravano, L., Zhai, C., Herzog, O., Evans, D. (eds.) Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management, Washington, DC, USA, November 8-13, 2004, pp. 652–659. ACM, New York (2004)
5. Hendler, J.: Web 3.0: Chicken farms on the semantic web. IEEE Computer 41(1), 106–108 (2008)
6. Iskold, A.: Semantic search: The myth and reality. ReadWriteWeb (2008), http://www.readwriteweb.com/archives/semantic_search_the_myth_and_reality.php
7. Karandikar, N.: Powerset vs. cognition: A semantic search shoot-out. GigaOM Tech News and Views (2008), http://gigaom.com/2008/06/07/powerset-vs-cognition-a-semantic-search-shoot-out/
8. Padgham, L., Winikoff, M.: Developing Intelligent Agent Systems: A Practical Guide. John Wiley & Sons, Chichester (2004)
9. Pokahr, A., Braubach, L.: The webbridge framework for building web-based agent applications. In: Dastani, M.M., El Fallah Seghrouchni, A., Leite, J., Torroni, P. (eds.) LADS 2007. LNCS (LNAI), vol. 5118, pp. 173–190. Springer, Heidelberg (2008)
10. Shafi, S.M., Rather, R.A.: Precision and recall of five search engines for retrieval of scholarly information in the field of biotechnology. Webology 2(2) (2005), http://www.webology.ir/2005/v2n2/a12.html
11. Weber, N.: Ontologien zur multiagentengestützten Suche - Einsatz in der Automobildomäne unter Verwendung von Jadex. Diplomarbeit, Distributed Systems and Information Systems Group, Computer Science Department, University of Hamburg (2009) (in German)
12. Wikipedia. Vertical search — wikipedia, the free encyclopedia (2009) (accessed May 6, 2009)
13. World Wide Web Consortium (W3C). SPARQL Query Language for RDF (January 2008)

Author Index