LNAI 5180

Matthias Klusch
Michal Pechoucek
Axel Polleres (Eds.)

Cooperative Information Agents XII

12th International Workshop, CIA 2008
Prague, Czech Republic, September 2008
Proceedings

Springer

Lecture Notes in Artificial Intelligence 5180

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

Subseries of Lecture Notes in Computer Science

Matthias Klusch   Michal Pechoucek
Axel Polleres (Eds.)

Cooperative Information Agents XII

12th International Workshop, CIA 2008
Prague, Czech Republic, September 10-12, 2008
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

Matthias Klusch
German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany
E-mail: klusch@dfki.de

Michal Pechoucek
Czech Technical University
Agent Technology Group
Karlovo namesti 13, 121 35 Prague 2, Czech Republic
E-mail: pechouc@labe.felk.cvut.cz

Axel Polleres
National University of Ireland
Digital Enterprise Research Institute
IDA Business Park, Lower Dangan, Galway, Ireland
E-mail: axel.polleres@deri.org

Library of Congress Control Number: 2008934044

CR Subject Classification (1998): I.2.11, I.2, H.4, H.2.8, H.3.3, C.2.4

LNCS Sublibrary: SL 7 – Artificial Intelligence

ISSN	0302-9743
ISBN-10	3-540-85833-4 Springer Berlin Heidelberg New York
ISBN-13	978-3-540-85833-1 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12512379 06/3180 5 4 3 2 1 0

Preface

These are the proceedings of the 12th international workshop on cooperative information agents (CIA 2008), held at the Czech Technical University in Prague, Czech Republic, on September 10–12, 2008.

In today's world of ubiquitously connected heterogeneous information systems, business services and computing devices, the intelligent coordination and provision of relevant added-value information or services to the user at any time, anywhere is of key importance to a variety of applications. This challenge is envisioned to be coped with by means of appropriate intelligent and cooperative information agents.

An intelligent information agent is a computational software entity that is capable of accessing one or multiple, potentially heterogeneous and distributed information sources, proactively acquiring, mediating, and maintaining relevant information or services on behalf of its human users, or other agents, preferably just in time and anywhere. One key challenge of developing intelligent and cooperative information systems is to balance the autonomy of networked data, information, and knowledge sources with the potential payoff of leveraging them by the appropriate use of such agents.

Research on intelligent information agents and systems is inherently cross disciplinary covering themes from domains such as AI, Multiagent Systems, HCI, Semantic Web Technologies, Web Services, Information Systems, Knowledge Discovery, Information Retrieval, and P2P Computing.

The objective of the international workshop series on cooperative information agents (CIA), since its establishment in 1997, has been to provide a distinguished, interdisciplinary forum for researchers, programmers, and managers to get informed about, present, and discuss the latest high-quality results in the research and development of agent-based intelligent and cooperative information systems, and applications for the Internet, Web and Semantic Web. Each event in the series offers regular and invited talks of excellence, given by renowned experts in the field, a selected set of system demonstrations, and honors innovative research and development of information agents by means of a best paper award and a system innovation award, respectively. The proceedings of the series are regularly published as volumes of Springer's Lecture Notes in Artificial Intelligence (LNAI) series.

In keeping with its tradition, this year's workshop featured a number of excellent regular and invited talks given by leading researchers covering a broad area of interest. In particular, CIA 2008 featured 5 invited and 19 regular papers selected from 38 submissions. The papers selected during the peer-review evaluation process are all included in this volume, which contains interesting, inspiring, and advanced work on the research and development of intelligent information agents worldwide. All workshop proceedings have been published by

Springer-Verlag as Lecture Notes in Artificial Intelligence volumes: 1202 (1997), 1435 (1998), 1652 (1999), 1860 (2000), 2182 (2001), 2446 (2002), 2782 (2003), 3191 (2004), 3550 (2005), 4149 (2006), 4676 (2007).

The CIA 2008 workshop issued a best paper award and a system innovation award to acknowledge and honor highly-innovative research and development, respectively, in the area of intelligent and cooperative information agents for the Internet and the Web. The system award was sponsored by Whitestein Technologies, the best paper award was sponsored by the workshop series. There has also been some financial support available to a limited number of students as (co-)authors of accepted papers to present their work at the CIA 2008 workshop; these grants were sponsored by IEEE FIPA standard committee, and the workshop series.

The CIA 2008 workshop was organized in cooperation with the Association for Computing Machinery (ACM), in particular the ACM special interest groups on Artificial Intelligence (SIGART), on Hypertext, Hypermedia and Web (SIGWEB), and on Knowledge Discovery in Data (SIGKDD). We are very grateful and indebted to our sponsors for the financial support that made this event possible. The sponsors of CIA 2008 were:

- Czech Technical University in Prague, Czech Republic
- IEEE Computer Society Standards Organisation Committee on Intelligent and Physical Agents (FIPA)
- Air Force Research Laboratory (AFRL), USA
- Whitestein Technologies, Switzerland
- Rockwell Automation, Czech Republic
- Digital Enterprise Research Institute (DERI) at the National University of Ireland, Galway
- Certicon, Czech Republic

We are also very grateful to the authors, the invited speakers, and attendees for contributing and discussing the latest results in relevant areas of this workshop, as well as to all members of the program committee, and the external reviewers for their critical reviews of submissions. Finally, a particularly cordial thanks goes to the local organization team from the Czech Technical University for providing an excellent venue and facilities and a very nice social program in the beautiful city of Prague.

We hope you enjoyed CIA 2008 and were inspired for your own work!

September 2008

Matthias Klusch
Michal Pechoucek
Axel Polleres

Organization

Co-chairs

Matthias Klusch	DFKI, Germany, *Workshop Chair*
Michal Pechoucek	CTU Prague, Czech Republic
Axel Polleres	DERI Galway, Ireland

Program Committee

Wolfgang Benn	TU Chemnitz, Germany
Felix Brandt	LMU Munich, Germany
Monique Calisti	Whitestein Technologies, Switzerland
Jorge Cardoso	U Madeira, Portugal
William Cheung	BU Hong Kong, Hong Kong
Philippe Cudre-Mauroux	MIT, USA
Frank Dignum	U Utrecht, The Netherlands
John Domingue	Open University, UK
Boi Faltings	EPF Lausanne, Switzerland
Michael Fink	TU Vienna, Austria
Vladimir Gorodetsky	SPIIRAS, Russia
Francesco Guerra	U Modena e Reggio Emilia, Italy
Manfred Hauswirth	DERI Galway, Ireland
Michael Huhns	U South Carolina, USA
Toru Ishida	U Kyoto, Japan
Catholijn Jonker	TU Delft, The Netherlands
Manolis Koubarakis	TU Crete, Greece
Ryszard Kowalczyk	Swinburne UT, Melbourne, Australia
Sarit Kraus	Bar-Ilan U, Israel
Victor Lesser	U Massachusetts, USA
Stefano Lodi	U Bologna, Italy
Werner Nutt	FU Bozen-Bolzano, Italy
Sascha Ossowski	U Rey Juan Carlos, Spain
Aris Ouksel	U Illinois at Chicago, USA
Massimo Paolucci	DoCoMo Euro Labs, Germany
Jeffrey Rosenschein	HU Jerusalem, Israel
Michael Rovatsos	U Edinburgh, UK
Heiko Schuldt	U Basel, Switzerland
Onn Shehory	IBM Haifa Research Lab, Israel
Katia Sycara	Carnegie Mellon U, USA
Walt Truszkowski	NASA Goddard Space Flight Center, USA

Rainer Unland	U Duisburg-Essen, Germany
Gottfried Vossen	U Muenster, Germany
Gerhard Weiss	SCCH, Austria
Frank van Harmelen	VU Amsterdam, The Netherlands

External Reviewers

Holger Billhardt
Stefan Dietze
Felix Fischer
Stefania Galizia
Paul Harrenstein
Thomas Krennwallner
Sebastian Leuoth
Raz Lin
Magdalena Ortiz
Annett Priemel
Martin Rehak
Joachim Schwieren
Frank Seifert
Vlad Tanasescu
Dmytro Tykhonov
Jiri Vokrinek
Inon Zukerman

Table of Contents

Enabling Networked Knowledge*

Stefan Decker and Manfred Hauswirth

Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway
IDA Business Park, Lower Dangan, Galway, Ireland

Abstract. Despite the enormous amounts of information the Web has made accessible, we still lack means to interconnect and link this information in a meaningful way to lift it from the level of information to the level of knowledge. Additionally, new sources of information about the physical world become available through the emerging sensor technologies. This information needs to be integrated with the existing information on the Web and in information systems which requires (light-weight) semantics as a core building block. In this position paper we discuss the potential of a global knowledge space and which research and technologies are required to enable our vision of networked knowledge.

1 What Is Networked Knowledge?

The wealth of information and services on today's information infrastructures like the Internet and the Web has significantly changed everyday life and has substantially transformed the way in which business, public and private interactions are performed. The economic and social influence of the Web is enormous, enabling new business models and social change, and creating wealth. However, we have barely scratched the surface of what information technology can do for society. The Web has enabled information creation and dissemination, but has also opened the information floodgates. The enormous amount of information available has made it increasingly difficult to find, access, present and maintain information. As a consequence, we are literally *drowning in information and starving for knowledge.* However, systematic access to knowledge is critical for solving today's problems – on individual and organisational as well as global levels.

Although knowledge is inherently strongly interconnected and related to people, this interconnectedness is not reflected or supported by current information infrastructures. The lack of interconnectedness hampers basic information management and problem-solving and collaboration capabilities, like finding, creating and deploying the right knowledge at the right time. Unfortunately, this is happening at a time when the problems humanity has to face are more difficult than ever (e.g., climate change, energy and resource shortages, or globalisation).

New methods are required to manage and provide access to the world's knowledge, for individual as well as collective problem solving. The right methods and tools for

* The work presented in this paper was supported (in part) by the Líon project supported by Science Foundation Ireland under Grant No. SFI/02/CE1/I131.

interconnecting people and accessing knowledge will contribute to solving these problems by making businesses more effective, scientists more productive and bringing governments closer to their citizens. Thus, the focus on *Enabling Networked Knowledge* is essential.

What is Networked Knowledge and why is it important?

Besides the creation of knowledge through observation, networking of knowledge is the basic process to generate new knowledge. Networking knowledge, can produce a piece of knowledge whose information value is far beyond the mere sum of the individual pieces, i.e., it creates new knowledge. With the Web we now have a foundational infrastructure in place enabling the linking of information on a global scale. Adding meaning moves the interlinked information to the knowledge level: Web + Semantics = Networked Knowledge. Knowledge is the fuel of our increasingly digital service economy (versus manufacturing economy); linking information is the basis of economic productivity.

Fortunately, current developments are helping to achieve these goals. Originating from the Semantic Web effort, more and more interlinked information sources are becoming available online, leading to islands of networked knowledge resources and follow-up industrial interest. Due to its forward-looking investment of the European Commission, Europe is playing a central and internationally recognised role in this development. Since more and more and different kinds of information sources become available, the main goal for the future is to build up on this leadership position.

As an example of a rapidly growing information space, Gartner predicts that *"By 2015, wirelessly networked sensors in everything we own will form a new Web. But it will only be of value if the 'terabyte torrent' of data it generates can be collected, analyzed and interpreted."* [1] Making sensor-generated information usable as a new and key source of knowledge will require its integration into the existing information space of the Web. Now is the time to tackle the next step: exploiting semantics to create an overall knowledge network bridging the islands enabling people, organisations and systems to collaborate and interoperate on a global scale, and bridging the gap between the physical world and the virtual world so that the information on the Web (the virtual world) can directly influence activities in the real world and vice versa. This integrated information space of networked knowledge will impact all parts of people's lives.

Hypothesis

It is our central hypothesis that collaborative access to networked knowledge assists humans, organisations and systems with their individual as well as collective problem solving, creating solutions to problems that were previously thought insolvable, and enabling innovation and increased productivity on individual, organisational and global levels.
In our opinion research needs to aim to:

$\boxed{\hookrightarrow}$

↪

1. *develop the tools and techniques for creating, managing and exploiting networks of knowledge;*
2. *produce real-world networks of knowledge that provide maximum gains over the coming years for human, organisational and systems problem solving;*
3. *validate the hypothesis; and*
4. *create standards supporting industrial adaptation.*

This overall research vision is broken down into three overall complementary research strands, which form the *Networked Knowledge House* (see Figure 1).

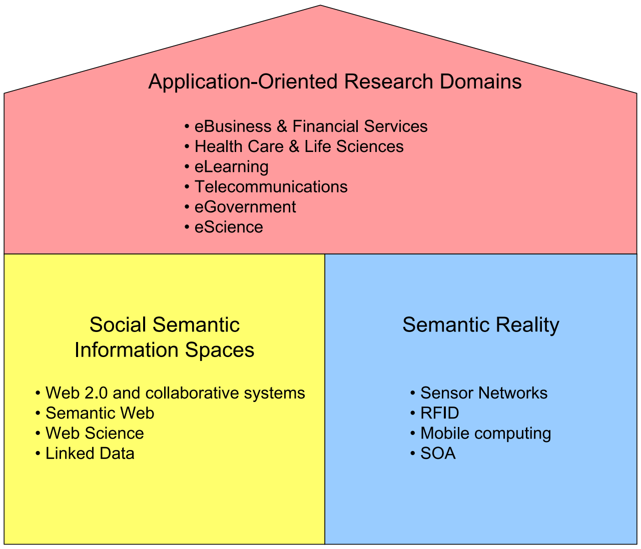

Application-Oriented Research Domains
• eBusiness & Financial Services
• Health Care & Life Sciences
• eLearning
• Telecommunications
• eGovernment
• eScience
Social Semantic Information Spaces
• Web 2.0 and collaborative systems
• Semantic Web
• Web Science
• Linked Data
Semantic Reality
• Sensor Networks
• RFID
• Mobile computing
• SOA

Fig. 1. Networked Knowledge House

Social Semantic Information Spaces deal with organization, linking, and management of knowledge on the Web. Semantic Reality addresses the integration of information from the physical world with knowledge in the virtual world (Social Semantic Information Spaces), the creation of knowledge out of information about the physical world, and efficient mechanisms to access this information at large scale via sensors. The technologies created by these basic research strands are then applied in and customized to a set of application domains, which we identified as most relevant to our work. This in turn requires research due to the specific requirements of the domains. Of course, the given list of application-oriented research domains is not comprehensive. A number of important domains are not listed, for example, environmental monitoring, traffic management and intelligent driving, logistics and tracking, or building management, to name a few, as they are beyond the scope of DERI at the moment.

In the following sections we explain the Networked Knowledge House in more detail.

2 Why Enabling Networked Knowledge?

The World Wide Web has dramatically altered the global communications and information exchange landscape, removing barriers of access, space and time from business and social transactions. The Web has created new forms of interaction and collaboration between systems, individuals and organisations. The dramatic development of the Web and the changes it has made to society are only a glimpse of the potential of a next-generation information infrastructure connecting knowledge and people. By interlinking the world's knowledge and providing an infrastructure that enables collaboration and focused exploitation of worldwide knowledge, Social Semantic Information Spaces and Semantic Reality, which will be explained in detail in the following sections, enable individuals, organisations and humanity as a whole to socialise, access services and solve problems much more effectively than we are able to today. The Web is already able to provide us with information, but lacks support for collaboration, knowledge sharing and social interaction. An information infrastructure supporting effective collaboration and augmented with interlinked and networked knowledge will support human capabilities and enable human-centric access to services and knowledge. We can already see the first glimpses of this in current online social networking sites (currently serving hundreds of millions of users), even though these sites are just data silos and do not interconnect knowledge efficiently.

Social Semantic Information Spaces and Semantic Reality as a networked knowledge infrastructure also make businesses more effective and scientists more productive by connecting them to the right people and to the right information at the right time and enabling them to recognise, collect, and exploit the relationships that exist between the knowledge entities in the world.

Vannevar Bush [2] and Doug Engelbart [3] were proposing similar infrastructures in 1945 and 1962. However, the technology available then was not advanced enough to realise their visions. Figuratively speaking, their ideas were proposing jet planes when the rest of the world had just invented the parts to build a bicycle. With the Semantic Web effort delivering standards to interconnect information globally and the Social Web showing how to collaborate on a global scale, now a window of opportunity has opened up to make these visions a reality and build a truly global networked knowledge infrastructure.

3 Social Semantic Information Spaces

One of the most visible trends on the Web is the emergence of "Social Web" (or Web 2.0) sites which facilitate the creation and gathering of knowledge through the simplification of user contributions via blogs, tagging and folksonomies, wikis, podcasts and the deployment of online social networks. The Social Web has enabled community-based knowledge acquisition, with efforts like Wikipedia demonstrating the "wisdom of the crowds" in creating the largest encyclopedia in the world. Although it is difficult to define the exact boundaries of what structures or abstractions belong to the Social Web, a common property of such sites is that they facilitate collaboration and sharing between millions of users. However, as more and more Social Web sites, communities and services come online, the lack of interoperation among them becomes obvious: the Social Web platforms create a set of isolated data silos – sites, communities and services that cannot interoperate with each other, synergies are expensive to exploit, and reuse and interlinking of data is difficult and cumbersome. The entities in the Social Web are not only data artefacts. Instead, it is a network of interrelated users and their concerns as well as content that the users are related to as producers, consumers or commentors. To enable machines to assist us with the detection and filtering of knowledge, many of these often implicit links have to be made explicit.

Social Semantic Information Spaces are a combination of the Semantic Web, the Social Web, collaborative working environments and other collaboration technologies. The goal behind Social Semantic Information Spaces is to create a universal collaboration and networked knowledge infrastructure, which interlinks all available knowledge and their creators. The resulting infrastructure would finally enable knowledge management capabilities as expressed by visionaries like Vannevar Bush and Doug Engelbart.

Figure 2 shows how Social Semantic Information Spaces fit into the current landscape: Communication and collaboration tools are augmented and made interoperable with Semantic Web technologies. The result is a network of accessible interlinked knowledge, enabling productive collaboration and knowledge management.

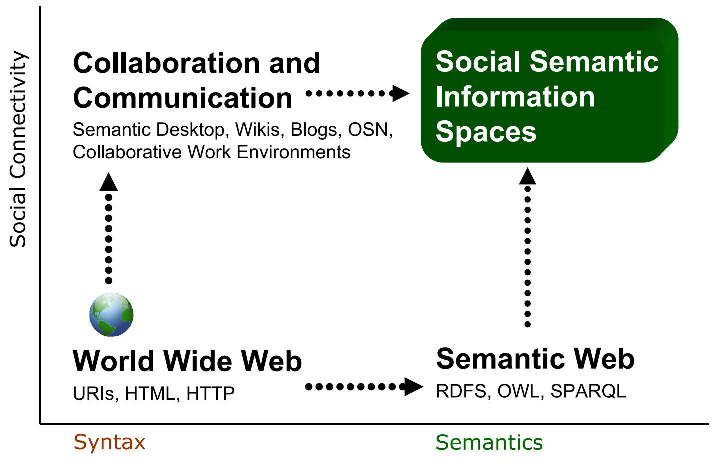

Social Connectivity
Collaboration and Communication
Semantic Desktop, Wikis, Blogs, OSN, Collaborative Work Environments
Social Semantic Information Spaces
World Wide Web
URIs, HTML, HTTP
Semantic Web
RDFS, OWL, SPARQL
Syntax
Semantics

Fig. 2. Social Semantic Information Spaces

In the following we list a couple of specific examples of enabling technologies and describe where and how they fit in the idea of a Social Semantic Information Space. All these technologies are just starting up and further research is necessary to ensure their development into broadly adopted technologies. However, convergence between some of the different efforts are already recognisable today.

3.1 Semantic Social Networks

From the beginning, the Internet was a medium for connecting not only machines but people. Email, mailing lists, the Usenet, and bulletin boards allowed people to connect and form online social networks, typically around specific topics. Although these groups did not explicitly define social networks, the ways people acted and reacted did so implicitly. The early Web continued this trend. More recently, sites such as Friendster and LinkedIn have brought a different notion of online communities by explicitly facilitating connections based on information gathered and stored in user profiles. However, all these sites are stovepipes and lock the information in: using the social network information for other purposes, e.g., for prioritising email as discussed in [4], requires standardised data exchange mechanisms. Initial crystallization points to remedy this situation are efforts like the Friend-of-a-Friend vocabulary (FOAF[1]) or the Semantically-Interlinked Online Communities initiative (SIOC[2] [5]). The SIOC initiative may serve as an example how social networking information can be interlinked with content such as online discussions taking place on blogs, message boards, mailing lists, etc. In combination with the FOAF vocabulary for describing people and their friends, and the Simple Knowledge Organization Systems (SKOS) model for organizing knowledge, SIOC enables the linkage of discussion postings to other related discussions, people (via their associated user accounts), and topics (using specific "tags" or hierarchical categories). As discussions begin to move beyond simple text-based conversations to include audio and video content, SIOC is evolving to describe not only conventional discussion platforms but also new Web-based communication and content-sharing mechanisms.

Some social networking sites, such as Facebook, are also starting to provide query interfaces to their data, which others can reuse and link to via the Semantic Web. Thus this information becomes part of the Web of information, which may be used or reused for a variety of purposes, providing crystallisation points for a network of knowledge.

3.2 Semantic Collaborative Technologies

Apart from the specific data representation mechanisms outlined above, other mechanisms and technologies contribute to the emergence of Social Semantic Information Spaces on the Web. The Social Semantic Desktop (SSD) [6] effort (materialised in the EU IP project NEPOMUK [7]) is aiming at organising information on the desktop by using Semantic Web metadata standards. Ontologies capture both a shared conceptualisation of desktop data and personal mental models. RDF serves as a common data representation format. As Web services can describe their capabilities and interfaces in a standardized way and become Semantic Web Services, on the desktop, applications, or rather their interfaces, can be modelled in a similar fashion. Together, these technologies provide a means to build the semantic bridges necessary for data exchange and application integration. The Social Semantic Desktop has the potential to transform the conventional desktop into a seamless, networked working environment, by obliterating the borders between individual applications and the physical workspace of different users.

[1] `http://www.foaf-project.org`
[2] `http://www.sioc-project.org`

In contrast to desktop applications Wikis have become popular Web-based collaboration tools and are widely deployed to enable organizing and sharing of knowledge. Wikis gained popularity since they enable the management of online content in a quick and easy way by "group-editing" using a simple syntax. However, typically knowledge collected in Wikis cannot be reused easily automatically and is only usable for human consumption. Semantic Web techniques applied to Wikis [8] leverage semantic technologies to address this challenge. They provide means to rapidly acquire formal knowledge also by non-knowledge engineers, and to create a network of this knowledge linked with other information sources. A typical example is the Semantic MediaWiki[3], which enables the evolution of Wikipedia into a reusable knowledge source enabling automatic processing and human support.

4 Semantic Reality

Until now the virtual world of information sources on the World Wide Web and activities in the real world have always been separated. However, knowledge accessible on the Web (the virtual world) may influence activities in the real world and vice versa, but these influences are usually indirect and not immediate. In contrast to this, imagine a world where:

- Cars know where the traffic jams are and traffic can be managed based on real-time input about the traffic situation.
- Medical data monitored through body sensor networks is automatically included into a patient's electronic healthcare record. Should a critical condition be detected by these sensors, the patient can be physically located and the closest doctor on duty can be guided to the patient, whilst preparing the necessary resources in the hospital the patient is to be transferred to.
- Your calendar knows how long the queue is at your physician.
- Your travel planner knows that the train is delayed before you go to the train station.
- Or generally, scarce resources can be managed efficiently and in-time.

The advent of sensor technologies in conjunction with the Semantic Web now provides the unique opportunity to unify the real and the virtual worlds as for the first time we have the necessary infrastructures in place or large-scale deployment will happen in the short term. Their combination will enable us to build very large information spaces and infrastructures which for the first time facilitate the information-driven online integration of the physical world and computers. Similarly, as the Internet has changed the way people communicate in the virtual world, Semantic Reality extends this vision to the physical world, enabling novel ways for humans to interact with their environment and facilitating interactions among entities of the physical world (Internet of Things). The physical world will be represented in cyberspace and information on our environment will become ubiquitously available on the Internet. This integrated information space has a wide range of applications in monitoring, manufacturing, health, tracking and planning.

[3] `http://meta.wikimedia.org/wiki/Semantic_MediaWiki`

We call it *Semantic* Reality because due to the possible scale and the overall goal of creating networked knowledge, understanding of information, i.e., semantics, plays a central role. Whether semantics is based on statistics, logical descriptions, or hybrid approaches does not matter. In fact, we believe that a wide spectrum of approaches and their combinations will be necessary to cover the diverse requirements. Semantic Reality aims at an integrated information space very much in line with the design philosophy of the original Internet, which embraces community-driven agreement processes, emergent behaviour and self-organisation, but adding semantics as a key enabling ingredient. Without machine-processable semantics such a large-scale system cannot work to its fullest extent. Yet, semantics must be light-weight, fault-tolerant, must support dynamic change, and has to be able to deal with incomplete, wrong, and noisy information in order to be applicable and useful in a global-scale heterogeneous environment. The rationale for success could be along the lines of "a little bit of semantics gets you a long way."

Figure 3 shows how Semantic Reality fits into the overall picture: Sensors connect the physical world to the computer, Social Semantic Information Spaces create and connect virtual worlds, and Semantic Reality integrates these two into one uniform information space which will provide novel ways of monitoring, controlling and influencing the environment, and how people and enterprises collaborate.

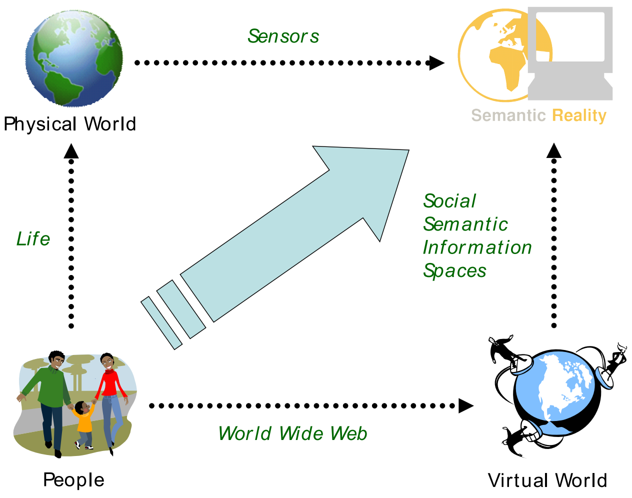


Fig. 3. Semantic Reality

The ultimate goal of Semantic Reality is "to deliver the right knowledge to the right people at the right time." This requires the adequate description of information, people and their requirements, and a temporal view on data sources, be they "real" or "virtual", i.e., a unified model of evolution of (integrated) information sources, thus moving from a static to a dynamic model of the Web and the physical world. This is currently taken into account only to a limited extent on the Web and only for "closed" applications, e.g., RSS feeds or blogs.

Semantic Reality shares several goals and properties with ubiquitous and pervasive computing and ambient intelligence. Though drawing on a large body of work in sensor networks, embedded systems, ubiquitous and pervasive computing, ambient intelligence, networking, distributed systems, distributed information systems, artificial intelligence, software engineering, social networking and collaboration, and Semantic Web, Semantic Reality is different from these research domains as it pushes the boundaries further by aiming at large-scale integration of (possibly isolated) information islands and the integration of systems, which requires the central use of semantics for information-driven integration and a uniform/universal, but light-weight semantic model of information sources and information.

The sheer size of the possible systems poses quite novel and unique challenges. Semantic Reality systems can only be built, deployed, and maintained if a large degree of self-organization and automatization capabilities are being built into the infrastructures and their constituents, enabling automated deployment (plug-and-play), automated (re-) configuration, automated component and information integration, and tailored information delivery based on user context and needs in a service-oriented way. The previous characteristics require semantic descriptions as a central ingredient: User requirements and contexts, the constituents of the system, the dynamic data (streams) they produce, their functionalities, and requirements – all need to be described using light-weight semantic mechanisms to enable a machine-understandable information space of real-world entities and their dynamic communication processes on a scale which is beyond the current size of the Internet.

In the following, we briefly discuss some of the core challenges and hint at possible strategies to address them.

Large-Scale and Open Semantic Infrastructures and Flexible Abstractions are required to enable the large-scale design, deployment and integration of sensor/actuator networks and their data. The integration has to happen on both the technical (data and network access) as well as on the semantic level ("What does the (stream) data provided actually mean?"). The infrastructure has to be open and easily extensible to address the heterogeneity issues which go far beyond those seen to date on the Internet. The infrastructure will draw on key enabling technologies such as (semantic) overlay networks using P2P technology to achieve scalability and light-weight semantic formats based on RDF and microformats. Middleware systems such as the Global Sensor Network (GSN) platform [9] are examples aiming at the development of a general-purpose middleware supporting these requirements. GSN is work-in-progress and provides a flexible middleware layer which abstracts from the underlying, heterogeneous sensor network technologies, supports fast and simple deployment and addition of new platforms, facilitates efficient distributed query processing and combination of sensor data, provides support for sensor mobility, and enables the dynamic adaption of the system configuration during runtime with minimal (zeroprogramming) effort. The GSN implementation is available from http://gsn.sourceforge.net/.

Query Processing, Reasoning, and Planning based on real-world sensor information will be core functionalities to exploit the full potential of Semantic Reality. The key research problems to overcome are the very large scale, the number of distributed information sources, the time-dependency of the produced data (streams), and the fact that the data is unreliable and noisy. For query processing this means to support distributed query processing and load-balancing at large scales with only incomplete views on the state of the overall system. In this context, distributed event-based infrastructures are of specific interest ("reactive" queries). Users should be able to register expressive, semantic "patterns" of interest and be notified by the system as soon as information satisfying their interests becomes available.

Also, new approaches for distributed reasoning and reasoning on time-dependant information, taking into account modalities and being based on an open-world assumption will be necessary. The size and the physical distribution of data will require new approaches combing logical and statistical approaches which will have to trade logical correctness with statistical guarantees and expressivity with scalability. Essentially, the goal is to enable "The World is the Database" scenarios with support for structured querying, integrated views (real-world information with virtual information), aggregation and analyses, and open, distributed reasoning over large, incomplete, and approximate data sets.

Cross-Layer Integration and Optimization will play a central role due to the extremely heterogeneous environment – a wide range of sensing devices with very heterogeneous hardware and processing characteristics; information systems and architectures along with virtual information streams which considerably increase complexity – and the various and often contradicting requirements on the different system levels. For example, sensor networks are optimized for life-time and offer only primitive programming and query models. If this is combined with the "wrong" distribution approach, e.g., a distributed hash table for discovery and the "wrong" distributed query processing approach which does not limit expressivity of queries, this will lead to an inefficient system design and limit the life-time of sensors by draining their power sources because of incompatible processing strategies at the different levels.

Semantic Description and Annotation of sensors, sensor data and other data streams will enable the flexible integration of information and (distributed) discovery of information. For scalability, integrity, and privacy reasons this has to be supported in a distributed fashion, for example, through semantic peer-to-peer systems. A prerequisite for discovery is the meaningful semantic description of sensors and sensor data by the manufacturer and by the user; for example, by the manufacturer through IEEE 1451 standard compliant Transducer Electronic Data Sheet (TEDS) [10], which essentially give a (non-semantic) description of the sensor that can very easily be ontologized, or via an ontologized subset of SensorML [11] which provides standard models and an XML encoding for describing sensors and measurement processes, and by the user by extending these basic descriptions with annotations adding more information, meaning, and links to other information.

Especially the annotation of sensor data itself will be highly relevant to understand the meaning of the produced data and share this knowledge. Visualization environments to support the annotation process will be of high importance. Such environments may support simple graphical annotation up to annotation with claims and findings in the case of scientific data. This derived knowledge then can be used again in the discovery process and will help to prevent "data graveyards" where interesting (measurement) information is available but cannot be used because the knowledge about its existence and meaning has been lost (the typical "PhD student finishes" syndrome). Due to the possibly large sizes of the produced data this poses additional scalability problems. As discovery, semantic annotation has to be supported in a distributed fashion, for example, by distributed semantic Wikis.

Emergent Semantics, Self-Organization, and Plug-and-Play are required to build working systems at the envisioned large scales where top-down system control, configuration, and enforcement of standards will be a very hard problem or even impossible. As we can see from the current community processes on the Web, a lot of successful defacto standards develop bottom-up. Conversely, these processes support the incremental development of standards and knowledge. The system must be able to self-organize and adopt its behavior in a plug-and-play fashion within organizational boundaries based on semantic understanding and agreement. Semantic understanding and agreements in turn will depend on dynamic processes which support (semi-)automatic assessment of the levels of agreement and their correctness. Such emergent semantic agreements can then be used as the basis for standardization (ontologies). Conversely, semantic formats can be advanced through such processes.

Semantically Enriched Social Network and Collaboration Infrastructures enable the targeted delivery of knowledge and information based on context description and actual user needs. The ubiquity of information requires means to filter and direct data streams on a need-to-know basis. The definition of user profiles, needs and contexts are key features enabling targeted information delivery and avoiding overload. Social networking information enables both – information sharing and information filtering based on interests and information needs.

Development Support and Tools along with experimental platforms and simulation tools will be necessary for efficient application development and testing. This means the availability of visual programming tools which support the developer in designing the acquisition, combination and integration of data. These designs then can be compiled down to the required target platforms (both sensor and back-end platforms, e.g., for business processes). To test applications, experimental testbeds along the lines of PlanetLab (http://www.planet-lab.org/) are essential as many of the characteristics of Semantic Reality systems require experimental evaluation under real-world conditions especially in terms of scale and distribution. To further evaluate applications, the integration of experiments and simulations should be supported in a seamless way, i.e., a test of an application in an experimental testbed should support the inclusion of simulation without changes to the application code. This means that parts of an application (or the complete application) should be able to run on an experimental testbed or on a simulator or any combination of those. On the application level modern paradigms such as service-oriented architectures, service mash-ups and Web 2.0-like functionalities should be available and be supported.

Integrity, Confidentiality, Reputation, and Privacy are the key security requirements for business users and consumers. The provided information has to be resistant against technical errors and attacks, has to be stored and transported in a secure way, has to come from authentic and trustworthy sources and must ensure the privacy of its providers and users. Physical distribution can be beneficial here as it helps to avoid the creation of "Big Brother" scenarios which consumers and legislators would not tolerate.

Vertical Integration of business processes ⇔ middleware ⇔ sensor/actuator networks relying on the above technologies and functionalities will then unleash the full potential of Semantic Reality. Sensor information, coming both from virtual and physical sources, are a key requirement for agile business processes requiring minimal human intervention.

5 Application-Oriented Research Domains

The Web has already influenced many different areas of society. The introduction of Social Semantic Information Spaces and Semantic Reality may have a similar influence, but like the Web, the transition of these new technologies into application areas is usually slow. To ensure rapid uptake and to provide maximum benefit to society, dissemination of research should focus on a number of carefully selected application research domains. These research domains investigate the adoption and uses of Social Semantic Information Spaces and Semantic Reality, combining a critical mass of technology-oriented research with the research on needs in specific application environments to initiate ground-breaking innovation. Example research domains are:

eHealth and Life Sciences: The objective of the eHealth and Life Sciences domain is to reduce the cost associated with the drug research and delivery process, making clinical research more efficient through data integration, and enabling patients' self-management of disease through telehealth, e.g., remote patient monitoring. Due to the heterogeneity of the eHealth domain, semantics is a crucial ingredient in achieving this objective.

eScience: The objective of the eScience domain is to improve collaboration among scientists working on computationally intensive problems, carried out in highly distributed network environments. Semantic support for distributed collaboration and annotation of scientific data and publications are of particular interest in our opinion.

Telecommunications: The objective of the telecommunications domain is to exploit semantic technologies for enabling telecoms to develop value-added communication services that will interface humans and machines, exploit machine-readable data, and improve intra-enterprise human communication and knowledge management. Context-information generated by sensors in conjunction with virtual information and unified communication profiles is of particular interest to enable new technology-supported communication paradigms.

eBusiness and Financial Services: The objective of the eBusiness and Financial Services domain is to apply new technology in the key areas of extracting business meaning from unstructured information, uncovering meaning within a business context, smarter Business Information Systems that can add meaning as they operate and communicate business information.

6 An Example Application Scenario

To illustrate the possibilities of “Networked Knowledge” we present a simple application scenario: Siobhan Hegarty who lives in Galway is pregnant with her second child. During her first pregnancy Siobhan has suffered from elevated blood sugar levels which can endanger the unborn child and the mother. The problem with elevated blood sugar levels in pregnant women is that the important characteristic which requires fast reaction is the change in the blood sugar level. Thus measuring it a few times a day is not sufficient but constant monitoring is required. Fortunately, mobile sensors are available which enable Siobhan to leave the hospital while her general practitioner (GP) stills get the relevant information. Siobhan is being equipped with a mobile blood sugar sensor which can transmit readings via Bluetooth. The device is paired with Siobhan’s mobile telephone which transmits the sensor readings via GSM. Additionally she gets a GPS device which records her position and sends it via her mobile.

Siobhan’s GP, Dr. James Mooney, enters the necessary monitoring requirements into his Care2X healthcare information system (`http://www.care2x.org/`) along with rules when to raise an alarm and to whom. For example, the system will call Siobhan and warn her via a synthesized message, while James is informed via a text message on his beeper which he wears all the time. The sensor readings from Siobhan’s blood sugar and GPS sensors are directly fed back into James’s Care2X system.

Let us assume that after some time, Siobhan’s blood sugar levels change dramatically and the alarm rules are set off. Now it is important to get Siobhan to a doctor as fast as possible, or vice versa – a doctor to Siobhan. Besides notifying Siobhan and James, the Care2X system accesses the information system of the hospital and requests a proposal, whether it is better to bring Siobhan into the hospital via an ambulance or bring a doctor to Siobhan. The hospital information system which knows the GPS position of all doctors with matching skills to help Siobhan and of all ambulances produces an optimal plan based on real-time sensor input from the traffic control system of the city. Given the current positions of available ambulances and doctors with the necessary skills, the optimal strategy is to pick up the endocrinologist Dr. Sarah O’Connor from her home with a nearby ambulance and bring her to Siobhan.

Unfortunately, while this plan was calculated two important changes to the scenario have happened: (1) No more readings from Siobhan’s GPS are received, probably because she has entered a building or because the device ran out of battery and Siobhan does not respond to calls on her mobile and (2) the last blood sugar readings show some strange and unknown pattern which neither James nor Sarah can interpret. As a reaction, the system now tries to locate Siobhan via other means: The system tries to determine her position via triangulation of her mobile and additionally informs all Bluetooth access points in the vicinity of her last position to send a message if they recognize any of her Bluetooth devices.

The strange patterns in the blood sugar readings worry James and Sarah and they decide to use their country-wide social network of clinical specialists to look for doctors who probably have already seen similar patterns. Additionally, they search medical databases on the Web for annotations describing such patterns. As a result of their search they find information which looks similar to the pattern they have seen but the result is inconclusive. In parallel, a colleague of them from Dublin who also participates in the social network they sent the symptoms to, informs them that the pattern may indicate a malfunction of the blood sugar sensor and describes his experiences.

In the meantime, Siobhan could be located by a Bluetooth access point. To be on the safe side the ambulance with Sarah on board is sent to her location and finds her in good condition. However, an examination reveals that indeed her blood sugar levels had changed dangerously and Siobhan is treated on the spot. After this successful intervention James and Sarah annotate the sensor readings to permanently store their findings. Their findings are stored in James's Care2X system, the hospital's information system and also made accessible to other doctors in the national infrastructure along with the actual sensor readings in a secure and anonymized way.

7 Core Research Topics for the Next Years

Core research objectives for the next years include the foundation for the creation of knowledge networks and collaboration infrastructures, which will support human capabilities and enable the human-centric access to services and knowledge on a global scale, opening up new opportunities for individuals and organisations. Example topics include:

Foundations for semantic collaboration: the development of technologies supporting distributed collaboration with a focus on the Semantic Desktop and the Web. Examples include APIs and ontologies that reuse existing social networking information from sites to assess the identity and relevance of information.

Scalable reasoning and querying facilities for knowledge: Current knowledge bases are not able to exploit and analyse knowledge, which would be necessary in order to learn from it. To exploit the available knowledge, scalable querying and data mining mechanisms need to be developed. Additionally, dynamic data sources (streams), modalities (time, space) and noise in the data, need to be taken into account and be supported.

Frameworks for semantic sensor networks: Currently sensor networks and the data they produce lack semantic description, making it difficult to integrate data coming from large-scale, dynamic sensor networks with existing information. It is necessary to develop practical semantic description methods for sensors and mobile device middleware, enabling the integration of sensor data with knowledge from knowledge networks. This will be part of a more general practical and deployable semantic service-oriented architecture.

8 Creating Impact

Knowledge networks are not created in a vacuum, but inside a highly dynamic information infrastructure – the Web, which provides us with a living laboratory enabling us to validate our approaches and hypothesis, and to improve our ideas.

The first way to validate the hypothesis is to study the usage of emerging networks of knowledge on the Web. Many application areas are dealing with the challenges of large, open, heterogeneous, dynamic and distributed environments. Semantics is an important cornerstone for achieving scalability of knowledge interchange and interoperability. Projects should validate this hypothesis by investigating the required research and approaches in application domains, ranging from eHealth to eGovernment to eLearning.

References

1. Raskino, M., Fenn, J., Linden, A.: Extracting Value From the Massively Connected World of 2015. Gartner Research (1 April 2005), http://www.gartner.com/resources/125900/125949/extracting_valu.pdf
2. Bush, V.: As We May Think. The Atlantic Monthly 176, 101–108 (1945)
3. Engelbart, D.C.: Augmenting Human Intellect: A Conceptual Framework. Stanford Research Institute, Menlo Park, CA, USA, Summary Report AFOSR-3233 (1962)
4. Golbeck, J., Hendler, J.: Reputation Network Analysis for Email Filtering. In: Conference on Email and Anti-Spam (CEAS), Mountain View, CA, USA (2004)
5. Breslin, J.G., Harth, A., Bojars, U., Decker, S.: Towards Semantically-Interlinked Online Communities. In: European Semantic Web Conference. Springer, Heidelberg (2005)
6. Decker, S., Frank, M.: The Social Semantic Desktop. Technical report, Digital Enterprise Research Institute (2004)
7. Groza, T., Handschuh, S., Moeller, K., Grimnes, G., Sauermann, L., Minack, E., Mesnage, C., Jazayeri, M., Reif, G., Gudjonsdottir, R.: The NEPOMUK Project - On the way to the Social Semantic Desktop. I-Semantics 2007, Journal of Universal Computer Science (2007)
8. Krötzsch, M., Vrandecic, D., Völkel, M.: Semantic MediaWiki. In: International Semantic Web Conference. Springer, Heidelberg (2006)
9. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. In: 8th International Conference on Mobile Data Management (2007)
10. NIST: IEEE1451 (2006), http://ieee1451.nist.gov/
11. Open Geospatial Consortium: Sensor Model Language (SensorML) (2008), http://vast.uah.edu/SensorML/

Coordination and Agreement in Multi-Agent Systems

Sascha Ossowski

Centre for Intelligent Information Technologies (CETINIA),
Universidad Rey Juan Carlos,
Calle Tulipán s/n,
28933 Móstoles (Madrid), Spain
sascha.ossowski@urjc.es

Abstract. It is commonly accepted that coordination is a key characteristic of multi-agent systems and that, in turn, the capability of coordinating with others constitutes a centrepiece of agenthood. However, the key elements of coordination models, mechanisms, and languages for multi-agent systems are still subject to considerable debate. This paper provides a brief overview of different approaches to coordination in multi-agent systems. It will then show how these approaches relate to current efforts working towards a paradigm for smart, next-generation distributed systems, where coordination is based on the concept of agreement between computational agents.

1 Introduction

Most current transactions and interactions at business level, but also at leisure level, are mediated by computers and computer networks. From email, over social networks, to virtual worlds, the way people work and enjoy their free time has changed dramatically in less than a generation time. This change has made that IT research and development focuses on aspects like new Human-Computer Interfaces or enhanced routing and network management tools. However, the biggest impact has been on the way applications are thought and developed. These applications require components to which more and more complex tasks can be delegated, components that show higher levels of intelligence, components that are capable of sophisticated ways of interacting, as they are massively distributed, sometimes embedded in all sort of appliances and sensors. In order to allow for an efficient design and implementation of systems of these characteristics, it is necessary to effectively enable, structure, and regulate their communications in different contexts.

Such an enterprise raises a number of technological challenges. Firstly, the open distributed nature of such systems adds to the *heterogeneity* of its components. The system structure may evolve at runtime, as new nodes may appear or disappear at will. There is also a need for on-the-fly alignment of certain concepts that interactions relate to, as the basic ontological conventions in such systems will be very limited. The *dynamicity* of the environment calls for a continuous *adaptation* of the structures that regulate the components' interactions, so as to achieve and sustain desired functional properties. But also non-functional issues related to *scalability*, *security*, and *usability* need to be taken into account. When designing mechanisms that address these challenges, the notion of *autonomy* becomes central: components may show complex patterns of activity aligned with the different goals of their designers, while it is usually impossible to directly influence their behaviour from the outside.

Coordination in multi-agent system (MAS) aims at harmonising the interactions of multiple autonomous components or agents. Therefore, it appears promising to review different conceptual frameworks for MAS coordination, and to analyse the potential and limitations of the work done in that field with regard to some of the aforementioned challenges.

This paper is organised as follows. Section 2 provides brief overview of coordination in MAS. Section 3 proposes the notion of *agreement* as a centrepiece of an integrated approach to coordination in open distributed systems, and outlines some research topics related to the vision of a technology of agreement. Some conclusions are drawn in Section 4.

2 Coordination in Multi-Agent Systems

Maybe the most widely accepted conceptualisation of coordination in the MAS field originates from Organisational Science. It defines coordination the *management of dependencies* between organisational activities [21]. One of the many workflows in an organisation, for instance, may involve a secretary writing a letter, an official signing it, and another employee sending it to its final destination. The interrelation among these activities is modelled as a *producer/consumer* dependency, which can be managed by inserting additional *notification* and *transportation* actions into the workflow.

It is straightforward to generalise this approach to coordination problems in multi-agent systems. The subjects whose activities need to be coordinated are the agents, while the entities between which dependencies are usually goals, actions or plans. Depending on the characteristics of the MAS environment, a taxonomy of dependencies can be established, and a set of potential coordination actions assigned to each of them (e.g.[36], [26]). Within this model, the *process* of coordination is to accomplish two major tasks: first, a *detection* of dependencies needs to be performed, and second, a *decision* respecting which coordination action to apply must be taken. A coordination *mechanism* shapes the way that agents perform these tasks [24].

The *result* of coordination, and its *quality*, is conceived differently at different levels of granularity. Understanding coordination as *a way of adapting to the environment* [36] is quite well suited to address this question from a *micro-level* (agent-centric) perspective. This is particularly true for multi-agent settings. If new acquaintances enter an agent's environment, coordination amounts to re-assessing its former goals, plans and actions, so as to account for the new (potential) dependencies between itself and other agents. If a STRIPS-like planning agent, for instance, is put into a multi-agent environment, it will definitely have to accommodate its individual plans to the new dependencies between its own prospective actions and potential actions of others, trying to exploit possible synergies (others may free certain relevant blocks for it), and avoiding harmful dependencies (making sure that others do not unstack intentionally constructed stacks etc). At this level, the result of coordination, the agent's adapted individual plan, is the better the closer it takes the agent to the achievement of its goals in the multi-agent environment.

From a *macro-level* (MAS-centric) perspective, the outcome of coordination can be conceived a “global” plan (or decision, action etc.). This may be a “joint plan” [29] if the agents reach an explicit agreement on it during the coordination process, or just the sum of the agents' individual plans (or decisions, actions etc. – sometimes called “multi-plan” [27]) as perceived by an external observer. Roughly speaking, the quality of the outcome of coordination at the macro-level can be evaluated with respect to the agents’ joint goals or the desired functionality of the MAS as a whole. If no such notion can be ascribed to the MAS, other, more basic features can be used instead. A good result of coordination, for instance, often relates to efficiency, which frequently comes down to the notion of Pareto-optimality. The amount of resources necessary for coordination (e.g. the number of messages necessary) is also sometimes used as a measure of efficiency.

The dependency model of coordination appears to be particularly well suited to *represent* relevant features of a coordination problem in MAS. The TAEMS framework [11], for instance, has been used to model coordination requirements in a variety of interesting MAS domains. It is also useful to rationalise observed coordination behaviour in line with a knowledge-level perspective [22]. Still, dependency detection may come to be a rather knowledge intensive task, which is further complicated by incomplete and potentially inconsistent local views of the agents. Moreover, making timely decisions that lead to efficient coordination actions is also everything but trivial. The problem becomes even more difficult when agents pursuing partially conflicting goals come into play [26]. In all but the simplest MAS, the instrumentation of these tasks gives rise to complex patterns of interactions among agents.

From a design perspective, coordination is probably best conceived as the effort of *governing the space of interaction* [6] of a MAS, as the basic challenge amounts to how to make agents converge on interaction patterns that adequately (i.e. instrumentally with respect to desired MAS features) solve the dependency detection and decision tasks. A variety of approaches that tackle this problem can be found in the literature, shaping the interaction space either directly, by making assumptions on agent behaviours and/or knowledge, or indirectly, by modifying the *context* of the agents in the MAS environment. The applicability of these mechanisms depends largely on the number and type of assumptions that one may make regarding the possibility of manipulating agent programs, agent populations, or the agents’ environment. This, in turn, is dependent on the characteristics of the coordination problem at hand.

The RICA-J framework [31], for instance, provides an ontology of interaction types, together with their associated protocols. Agents can freely choose to play or abandon certain roles within an interaction but, when using the framework, an agent programmer is limited to using protocol compliant actions.

Governing coordination infrastructures make a clear separation between the *enabling* services that they provide (e.g. communication channel or blackboard-based communication primitives) and the *governing* aspects of interaction, which are usually described within a declarative language (e.g. programmable tuple spaces) [25]. The access regulations for the elements of the MAS environment (resources, services, etc) expressed in such a language are sometimes called *environment laws* [30].

Electronic Institutions (EI) [23] use organisational abstractions to shape the interactions of the agents participating in them. Agents play different roles in the (sub-) protocols that, together with additional rules of behaviour, determine the legal sequences of illocutions that may arise within a particular instance of a scene. Scenes, in turn, are interconnected and synchronised by means of transitions within a performative structure. Norms, as additional institutional abstractions, express further behaviour restrictions for agents. In the EI framework, agents can only interact with each other through specific institutional agents, called governors [13], which assure that all behaviour complies with the norms and that it obeys the performative structure. So, different from the aforementioned approaches, the governing or regulating responsibility is transferred from the infrastructure to specialized middle agents.

From the point of view of an individual agent, the problem of coordination essentially boils down to finding the sequence of actions that, given the regulations within the system (or, if possible in a certain environment, the expected cost of transgressing them), best achieves its goals. In practice, this implies a series of non-trivial problems. Models of coalition formation determine when and with whom to form a team for the achievement of some common (sub-) goal, and how to distribute the benefits of synergies that arise from this cooperation [32]. Distributed planning approaches [12] may determine how to (re-)distribute tasks among team members and how to integrate results. From an individual agent's perspective, the level of trustworthiness of others is central to almost every stage of these processes, so as to determine whether other agents are likely to honour the commitments that have been generated [33].

An appealing way to tackle both the system-level and the agent-level requirements is to take an organisation-oriented tack towards the problem of MAS coordination. Organisational models underlying approaches such as Agent-Group-Role [14], MOISE [18], EI [23], or RICA [31] provide a rich set of concepts to specify and structure mechanisms that govern agent interactions through the corresponding infrastructures or middleware. But they can also facilitate the agents' local decision-making tasks. For instance, role and interaction taxonomies can be used to find suitable interactions partners, by providing additional information regarding the usability of services in a certain interaction context [15]. Structural information about roles can also be used for the bootstrapping of reputation mechanism, when only very limited information about past interactions is available in the system [5]. Role hierarchies, and other types of structural information, can also be extended on-the-fly to improve system performance [17]. In general, the fact that organisational structures may dynamically evolve, shifts the attention from their traditional use as a *design-time* coordination mechanism for mainly closed distributed problem-solving systems, to an adaptive *run-time* coordination mechanism also applicable to open MAS [24].

3 Towards a Technology of Agreement

The previous section has given a brief overview of work on coordination mechanisms that has been carried in the MAS field. Even though an attempt has been made to structure and present it in some coherent manner, the reader will have noticed that several quite different approaches and mechanisms coexist under the "umbrella" of the term coordination. Not all of them are relevant to the challenges for the design of open distributed systems outlined in the introduction. For instance, the whole set of *coupled* coordination mechanisms [35] are effectively useless for the purpose of this paper, as they require having a direct influence on the agent programs. On the other hand, the problem of semantic interoperability is usually outside the scope of MAS coordination models and languages.

The notion of *agreement* among computational agents appears to be better suited as the fundamental notion for the proposal outlined in this paper. Until recently, the concept of agreement was a domain of study mainly for philosophers, sociologists and was only applicable to human societies. In recent years, the growth of disciplines such as social psychology, socio-biology, social neuroscience, together with the spectacular emergence of the information society technologies, have changed this situation. Presently, agreement and all the processes and mechanisms implicated in reaching agreements between different kinds of agents are a subject of research and analysis also from technology-oriented perspectives.

The process of agreement-based coordination can be designed based on two main elements:

(1) a normative context, that determines the rules of the game, i.e. interaction patterns and additional restrictions on agent behaviour; and
(2) a call-by-agreement interaction method, where an agreement for action between the agents that respects the normative context is established first; then actual enactment of the action is requested.

The techniques based on organizational structures discussed in the previous section will be useful to specify and design such systems. In addition, semantic alignment, norms, argumentation and negotiation, as well as trust and reputation mechanisms will be in the "agreement technology sandbox".

Semantic technologies constitute a centrepiece of the approach as semantic problems pervade all the others. Solutions to semantic mismatches and alignment of ontologies [4] are needed to have a common understanding of norms or of deals, just to put two examples. The use of semantics-based approaches to service discovery and composition will allow exploring the space of possible interactions and, consequently, shaping the set of possible agreements [15].

At system-level, *norms* are needed to determine constraints that the agreements, and the processes to reach them, have to satisfy. Reasoning about a system's norms is necessary at design-time to assure that the system has adequate properties, but it may also be necessary at run-time, as complex systems usually need dynamic regulations [16]. *Organisational* structures further restrict the way agreements are reached by fixing the social structure of the agents: the capabilities of their roles and the relationships among them (e.g. power, authority) [3].

Moving further towards the agent-level, *negotiation* methods are essential to make agents reach agreements that respect the constraints imposed by norms and organisations. These methods need to be complemented by an argumentation-based approach: by exchanging arguments, the agents' mental states may evolve and, consequently, the status of offers may change [2] [7]. Finally, agents will need to use *trust* mechanisms that summarise the history of agreements and subsequent agreement executions in order to build long-term relationships between the agents. Trust is the technology that complements traditional security mechanisms by relying on social mechanisms that interpret the behaviour of agents [34].

One may conceive the aforementioned topics in a "tower structure", with semantic technologies at the bottom layer and trust mechanisms at the top, where each level provides functionality to the levels above [1]. Notice, however, that there is also a certain feedback from higher to lower layers as, for instance, reputation mechanisms may influence organisational structures such as role and interaction hierarchies [17]; and this information can as well be used for semantic alignment [4] and discovery [15].

4 Discussion

This paper has presented an overview of different approaches to coordination in the MAS field. It has been argued that the notion of agreement is essential to instil coordination in open distributed systems. Some existing technologies from the field of MAS coordination can be applied to this respect, but others – and in particular semantic technologies – need to be added. Several research efforts are currently ongoing that may contribute to the development of a "technology of agreement" in one or another way. The attempt to harmonise these efforts, which is currently being carried out at European level, promotes the emergence of a new paradigm for next generation distributed systems based on the notion of *agreement* between computational agents [9].

Acknowledgements

Many ideas reported in this paper draw upon joint work with Carles Sierra, Vicent Botti, and others, in the framework of a Spanish national project on "Agreement Technology". This term was first mentioned by Mike Wooldridge in internal discussions at the AAMAS conference in 2004. It has also been used as a title for a conference by Nick Jennings. I am also thankful to Axel Polleres, Cristiano Castelfranchi and Leila Amgoud for their comments regarding the different technological challenges related to a computational notion of agreement, as well as Andrea Omicini for our discussions on the different stances on coordination in multi-agent systems. This work was partially supported by the Spanish Ministry of Science and Innovation, grants TIN2006-14630-C03-02 and CSD2007-00022 (CONSOLIDER-INGENIO 2010).

References

[1] Agreement Technologies project homepage, `http://www.agreement-technologies.org/`

[2] Amgoud, L., Dimopolous, Y., Moraitis, P.: A unified and general framework for argumentation-based negotiation. In: Proc. 6th Int. Joint Conference on Autonomous Agents and Multi-Agents Systems (AAMAS 2007), pp. 963–970. IFAAMAS (2007)

[3] Argente, E., Julian, V., Botti, V.: Multi-Agent System Development based on Organizations. Electronic Notes in Theoretical Computer Science 150(3), 55–71 (2006)

[4] Atienza, M., Schorlemmer, M.: I-SSA - Interaction-situated Semantic Alignment. In: Proc Int. Conf. on Cooperative Information Systems (CoopIS 2008) (to appear, 2008)

[5] Billhardt, H., Hermoso, R., Ossowski, S., Centeno, R.: Trust-based Service Provider Selection in Open Environments. In: Proc. ACM Symposium on Applied Computing (SAC-2007), pp. 1375–1380. ACM Press, New York (2007)

[6] Busi, N., Ciancarini, P., Gorrieri, R., Zavattaro, G.: Coordination Models - A Guided Tour. In: Omicini, et al. (eds.) Coordination of Internet Agents: Models, Technologies, and Applications, pp. 6–24. Springer, Heidelberg (2001)
[7] Caminada, M., Amgoud, L.: On the evaluation of argumentation formalisms. Artificial Intelligence Journal 171(5-6), 286–310 (2007)
[8] Castelfranchi, C., Dignum, F., Jonker, C., Treur, J.: Deliberative Normative Agents - Principles and Architecture. In: Jennings, Lespérance (eds.) Intelligent Agents VI. LNCS, vol. 1757, pp. 364–378. Springer, Heidelberg (2000)
[9] COST Act. IC0801, http://www.cost.esf.org/index.php?id=110&action_number=IC0801
[10] Debenham, J., Sierra, C.: Merging intelligent agency and the Semantic Web. Knowledge-Based Systems 21(3), 184–191 (2008)
[11] Decker, K.: TAEMS: A Framework for Environment Centered Analysis and Design of Coordination Mechanisms. In: O'Hare, Jennings (eds.) Foundations of Distributed Artificial Intelligence. John Wiley and Sons, Chichester (1996)
[12] Durfee, E.: Distributed Problem Solving and Planning. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001 and EASSS 2001. LNCS (LNAI), vol. 2086, pp. 118–149. Springer, Heidelberg (2001)
[13] Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: AMELI - An agent-based middleware for electronic institutions. In: Proc. of the Third Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004), pp. 236–243. ACM Press, New York (2004)
[14] Ferber, J., Gutknecht, O., Fabien, M.: From Agents to Organizations - An Organizational View of Multi-agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
[15] Fernández, A., Ossowski, S.: Exploiting Organisational Information for Service Coordination in Multiagent Systems. In: Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2008), pp. 257–264. IFAAMAS (2008)
[16] Gaertner, D., García-Camino, A., Noriega, P.,Rodríguez-Aguilar, J.A., Vasconcelos, W.: Distributed norm management in regulated multiagent systems. In: Proc. Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2007), pp. 624–631. IFAAMAS (2007)
[17] Hermoso, R., Centeno, R., Billhardt, H., Ossowski, S.: Extending Virtual Organizations to improve trust mechanisms (Short Paper). In: Proc. of the Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS-2008), pp. 1489–1492. IFAAMAS (2008)
[18] Hubner, J., Sichman, J., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. Int. Journal of Agent-Oriented Software Engineering 1(3/4), 370–395 (2006)
[19] Klusch, M., Sycara, K.: Brokering and matchmaking for coordination of agent societies: a survey. In: Coordination of Internet Agents: Models, Technologies, and Applications (Omicini y otros), pp. 197–224. Springer, Heidelberg (2001)
[20] Klusch, M., Fries, B., Sycara, K.: Automated Semantic Web Service Discovery with OWLS-MX. In: Proceedings of 5th International Conference on Autonomous Agents and Multi- Agent Systems (AAMAS-2006), pp. 915–922. ACM Press, New York (2006)
[21] Malone, T., Crowston, K.: The Interdisciplinary Study of Co-ordination. Computing Surveys 26(1), 87–119 (1994)
[22] Newell, A.: Reflections on the Knowledge Level. Artificial Intelligence 59, 31–38 (1993)
[23] Noriega, P., Sierra, C.: Electronic Institutions – Future Trends and Challenges. In: Klusch, M., Ossowski, S., Shehory, O. (eds.) CIA 2002. LNCS (LNAI), vol. 2446, pp. 14–17. Springer, Heidelberg (2002)

[24] Omicini, A., Ossowski, S.: Objective versus Subjective Coordination in the Engineering of Agent Systems. In: Klusch, M., Bergamaschi, S., Edwards, P., Petta, P. (eds.) Intelligent Information Agents. LNCS (LNAI), vol. 2586, pp. 179–202. Springer, Heidelberg (2003)

[25] Omicini, A., Ossowski, S., Ricci, A.: Coordination Infrastructures in the Engineering of Multiagent Systems. In: Bergenti, Gleizes, Zambonelli (eds.) Methodologies and software engineering for agent systems – The Agent-Oriented Software Engineering Handbook, pp. 273–296. Kluwer, Dordrecht (2004)

[26] Ossowski, S.: Co-ordination in Artificial Agent Societies. LNCS (LNAI), vol. 1535. Springer, Heidelberg (1998)

[27] Ossowski, S.: Constraint Based Coordination of Autonomous Agents. Electronic Notes in Theoretical Computer Science 48, 211–226 (2001)

[28] Ossowski, S., Menezes, R.: On Coordination and its Significance to Distributed and Multi-Agent Systems. Journal of Concurrency and Computation - Practice and Experience 18(4), 359–370 (2006)

[29] Rosenschein, J., Zlotkin, G.: Designing Conventions for Automated Negotiation. AI Magazine 15(3), 29–46 (1995)

[30] Schumacher, M., Ossowski, S.: The governing environment. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2005. LNCS (LNAI), vol. 3830, pp. 88–104. Springer, Heidelberg (2006)

[31] Serrano, J.M., Ossowski, S.: On the Impact of Agent Communication Languages on the Implementation of Agent Systems. In: Klusch, et al. (eds.) Cooperative Information Agents VIII. LNCS, vol. 2782, pp. 92–106. Springer, Heidelberg (2004)

[32] Shehory, O., Sycara, K., Somesh, J.: Multi-agent Coordination through Coalition Formation. In: Rao, A., Singh, M.P., Wooldridge, M.J. (eds.) ATAL 1997. LNCS, vol. 1365, pp. 143–154. Springer, Heidelberg (1998)

[33] Sabater, J., Sierra, C.: Review on Computational Trust and Reputation Models. Artificial. Intelligence Review 24(1), 33–60 (2005)

[34] Sierra, C., Debenham, J.: Information-Based Agency. In: Proc Int. Joint Conference on AI (IJCAI-2007), pp. 1513–1518. AAAI Press, Menlo Park (2007)

[35] Tolksdorf, R.: Models of Coordination. In: Omicini, A., Tolksdorf, R., Zambonelli, F. (eds.) ESAW 2000. LNCS (LNAI), vol. 1972. Springer, Heidelberg (2000)

[36] von Martial, F.: Coordinating Plans of Autonomous Agents. LNCS (LNAI), vol. 610. Springer, Heidelberg (1992)

Agents and Databases: A Symbiosis?

Heiko Schuldt

Database and Information Systems Group
Department of Computer Science
University of Basel, Switzerland
heiko.schuldt@unibas.ch

Abstract. Over the last decades, data and information management has been subject to significant changes. Access to data and information is no longer provided by monolithic database systems. Rather, applications need to cope with an increasing number of heterogeneous and distributed data and information sources, ranging from traditional databases, large document collections and information sources on the Internet and the Semantic Web. This also affects the way data and information is searched, accessed, and processed. In particular, the agent community has addressed this change and has spawned the field of *information agents*. An information agent pro-actively searches, retrieves, accesses and maybe even processes information on behalf of its user. Also the database community has faced the challenges stemming from this change by making database functionality available even outside of database systems.

In this paper, we review the recent developments in both fields and show examples of activities which lead to synergies in both communities and which emphasize on the potential for symbiotic co-existence.

Keywords: Cooperative Information Agents, Databases, Hyperdatabase Systems, Agents and Transactions.

1 Introduction

Over the last decades, data and information management has undergone considerable changes. From rather monolithic, database-centric applications where access to data was directly provided by (mostly relational) database management systems (DBMSs), the evolution first led to an increasing number of heterogeneous and distributed data and information sources, ranging from traditional databases and large (multimedia) document collections to information sources on the Internet, and finally to information in the Semantic Web and even to embedded information sources in mobile "smart" objects as they occur in a pervasive computing environment. This development significantly affects the way data and information is searched, accessed, and processed. Both the immense amount of information and the number of different information sources poses a great challenge for appropriate infrastructures for dealing with search, access, management, and processing. In particular, the agent community has addressed this change and has spawned the field of *information agents* [13,9]. A cooperative information agent is a *computational software entity that has access to one or multiple, potentially heterogeneous, and geographically and logically distributed data and information sources, pro-actively acquires, mediates, and maintains relevant information on behalf of its human users or other agents, preferably just-in-time* [4]. Several workshop and conference series have provided the fora necessary for creating a pertinent community and have helped the field in coming of age.

At the same time, database systems have evolved and the database community has also faced the challenges stemming from this change. In particular, databases are more and more hidden behind service interfaces. At the same time, database concepts such as query processing and transactions are provided outside of database systems, at the level of service invocations.

In this paper, we review the recent developments in both fields. In particular, we aim at answering the rather rhetorical question raised in the title of this paper by identifying areas of mutual interest and activities which hopefully lead to synergies in both communities. The hyperdatabase vision [16] will be presented as one example for activities in the intersection between both areas and we describe in more detail two concrete realizatons of this vision to exemplify the relationship between both fields and to stress the possibility for symbiotic co-existence.

The paper is organized as follows: Section 2 briefly introduces the two different fields and identifies the potential for cross-fertilizations. In Section 3, we illustrate the possible symbiosis between information agents and databases by presenting the hyperdatabase vision. In particular, we present two selected hyperdatabase implementations, namely OSIRIS which provides optimized routing of service requests for distributed processes orchestrated by means of cooperating agents and AMOR which provides transactional execution guarantees for cooperating agents. Finally, Section 4 concludes.

2 Information Agents and Databases

In what follows, we first give a brief introduction on information agents and databases, review their development and analyze the relationship between both fields.

2.1 Information Agents

In short, the main task of information agents can be summarized as providing integrated access to information from potentially heterogeneous information sources hosted at several locations. The activities of information agents include the discovery of information sources, the integration of information from different sources and possibly also the processing of information, e.g., to derive new information [13,9].

In more detail, according to the terminology defined by the AgentLink Special Interest Group on Intelligent Information Agents [2], an information agent is a computational software entity that may access one or multiple, distributed and heterogeneous information sources, and pro-actively acquires, mediates, and maintains relevant information on behalf of its user(s) or other agents preferably just-in-time. This includes their ability to semantically broker information by providing a pro-active resource discovery, by mediating between information consumers and providers and finally by offering value-added information services and products to the user or other agents. The latter implies that information agents also act as producers of information and not just facilitators for accessing information, and thus have to take care of the quality of their services. Information agents take over the role of brokers between information sources (e.g., by accessing databases), other agents (or their services, respectively), and human users.

Different information agents may cooperate in order to jointly achieve a common task. Furthermore, they may be subject to dynamic re-configurations, e.g., to react to changes in their environment react or to increase the quality of their service (load balancing, reduction of data transfer, etc.).

2.2 Databases

Relational database systems have been introduced more than thirty years ago. They have been considered as infrastructure and main platform for development of data-intensive applications. The notion of "data independence", part of Codd's rules [5], was a breakthrough because programmers were freed from low-level details, e.g., how to access shared data efficiently and correctly, given concurrent access. But already in the late nineties, the prerequisites for application development have changed dramatically. Storage and communication has become fairly cheap, and the internet has started to dominate modern information infrastructures. Consequently, the role of database concepts had to be re-visited and newly determined. Undoubtedly, the database system has played and still plays an important role. However, it has more and more degenerated to a storage manager, far away from the applications. In this situation, about a decade ago, researchers started to question the role of databases for future distributed information systems engineering ("Databases Breaking out of the Box" [25] and the Lowell Database Research Self-Assessment Report [1]).

One of the consequences of this development is that databases are increasingly becoming invisible, although they still constitute the backend-tier of data-intensive applications. Rather, data management and thus databases are hidden behind service interfaces. Thus, data-intensive applications have to deal with services description and registration instead of relational schema definition, service instances instead of relations, or service invocations instead of relational operators for accessing or manipulating data. Database research thus more and more needs to cope with database functionality at higher levels of semantics, e.g., optimal routing of service requests as opposed to query optimization, or transactional execution guarantees for composite services to just name a few.

2.3 Symbionts or Predators, Peaceful Coexistence or Mutual Indifference?

In light of these developments, it is obvious that the relationship between information agents and databases cannot be characterized by just mutual indifference.

It even significantly goes beyond a peaceful coexistence since information agents are more than only clients to databases.

From the point of view of information agents, databases are still the resource managers which persistently store information. But even more, the broader notion of databases provides the necessary protocols and mechanisms for coordinating agent interactions, e.g., for providing provably correct and transactionally safe multi-agent executions or for routing service requests in an optimal way.

At the same time, higher level database functionality can significantly benefit from advances in the area of information agents. This includes the ability to semantically integrate information from different sources, to negotiate quality of service for interactions, to proactively discover resources, or to dynamically adapt to changing environments.

Apparently, there are many potential synergies between both fields. Some of them are addressed in recent initiatives and projects which are stemming from the database community and which aim at making database functionality available outside of databases. Thus, these projects implicitly address issues which are of high practical impact also for information agent interactions. Hyperdatabases [15,16,17], InfoSphere [14], or AutoGlobe/ServiceGlobe [6] are some examples out of a longer list of similar initiatives which hopefully lead to a sustainably symbiotic relationship between both fields.

In what follows, we will present in detail the hyperdatabase project which has originated at ETH Zurich and which is now being continued at the University of Basel. In particular, we briefly present the underlying hyperdatabase vision for the management of future information spaces and we will present two implementations of the hyperdatabase vision which provide database functionality at higher level of abstractions, outside of a database, which is supposed to also facilitate and ameliorate distributed agent-based applications.

3 Hyperdatabases

This section first briefly introduces the hyperdatabase vision and then presents two implementations of this vision for which we believe they will have a strong impact on the way distributed information agents interact. A more detailed summary of the hyperdatabase vision and its realizations can be found in a recent survey paper [16].

3.1 The Hyperdatabase Vision

The driving forces behind the hyperdatabase vision are mainly based on two observations. First, that the volume of data and information is significantly increasing and undergoes continuous changes while being inherently distributed and heterogeneous. Second, non-relational data sources such as, for instance, image, video, audio collections are increasingly gaining importance. Thus, a holistic approach to managing the “information space” of the future, i.e., the universe of all information sources, needs a radical departure from the traditional database thinking by moving up to a much higher level of abstraction. In short, a hyperdatabase administers objects that are composed of objects and transactions that are composed of transactions. Thus, it provides database functionality not only over many distributed databases but in a more general way on top of distributed components and services with various functionality in a networked environment.

With hyperdatabases, the notion of data independence is generalized in the form of "higher order data independence". This includes the immunity of application programs not only against changes in storage and access structure, but also against changes in location, implementation, workload, the number of replica of software components and their services. The relation between databases and hyperdatabases can be briefly characterized as follows: a database is a platform for clients concurrently accessing shared data which needs data definition, data manipulation, and transactions at the interface. Internally, the database management system performs query optimization, provides correctness for parallel access, recovery, persistence, load balancing, and guarantees a high degree of availability. Similarly, a hyperdatabase is a platform for clients, concurrently accessing shared application services; thus, as opposed to shared data in a database, it has to deal with shared components and services. At the interface, a hyperdatabase has to provide component and service definition and description, service customization, transactional processes encompassing multiple service invocations. Internally, the hyperdatabase performs optimization of client requests, routing, scheduling, and parallelization, correctness of concurrent accesses, flexible failure treatment, providing guaranteed termination (i.e., a generalized form of atomicity), availability, flexible recovery, and scalability. Table 1 summarizes the analogy.

Most importantly and in contrast to traditional database technology, a hyperdatabase infrastructure must not follow monolithic system architecture but must be fully distributed over all participating nodes in a network. Every node is equipped with an additional thin software layer, a so-called hyperdatabase layer (which, in the terminology of the agent community, is actually an information agent). Each deployment of the hyperdatabase layer can be considered as an agent which offers dedicated services and/or provides access to information. Thus, one of the main challenges of hyperdatabases is the communication and coordination of these hyperdatabase layers.

3.2 Hyperdatabase Projects

The following two sections present in more detail two concrete implementations which arose from the hyperdatabase vision. Both address the distributed retrieval and/or processing of information by a set of cooperating hyperdatabase layers (agents). The first focuses on distributed, process-based applications while the second addresses transactional semantics in these distributed settings.

OSIRIS: A Hyperdatabase Implementation for Distributed Process-based Applications. The proliferation of service-oriented computing and in particular of (Web) services had a strong impact on information systems. System

Table 1. Analogy between DBMSs and the Hyperdatabases (from [16])

Database Management	**Hyperdatabase Infrastructure**
Relational schema definition	Service definition and registration
Relational schema extension	New service registration
Relation	Service instance
Access to relation	Service invocation
Query and update language	Process definition language
Transaction	Transactional process
ACID Guarantees	Correct execution and guaranteed termination of process
Undo operation	Inverse service invocation
Redo operation	Repeatable service invocation
Indexing	Feature extraction and feature space organization
Query Optimization	Optimal process routing
Physical Database Design	Configuration Design by service allocation and replication

support for the invocation of single services is widely available, due to standardized protocols and formats (e.g., WSDL and SOAP). Beyond these basics, the most important challenges are the management of existing services and their evolution, the composition of existing services into a coherent whole by means of processes, and the optimization of service requests to guarantee a high degree of scalability in order to deal with an increasing number of services, processes, and users.

OSIRIS (Open Service Infrastructure for Reliable and Integrated process Support) [22,23] is a novel infrastructure for distributed service-oriented applications. It primarily focuses on the scalable and reliable execution of composite services, also called processes. OSIRIS provides basic mechanisms which can also be applied to distributed, cooperating information agents in jointly achieving a common task. Process-based applications are either explicitly specified, according to the paradigm of programming in the large [26], or are individually and automatically created by a dedicated planner (e.g., [10,11,24]).

OSIRIS consists of a set of agents (so-called hyperdatabase layers). These agents interact in a decentralized, peer-to-peer style for executing processes [20]. In addition, OSIRIS considers several global repositories. While only the agents are responsible for process execution, the global repositories collect metadata on the overall system and apply sophisticated replication mechanisms (based on publish/subscribe techniques) for control flow dependencies from the global repositories to the agents. At run-time, this guarantees that no single point of failure is involved in the execution of processes and allows to provide sophisticated load balancing strategies (selection of the least loaded peer which provides a dedicated service). For this, the concrete service binding is determined at run-time depending on the load of agents and costs of invoking a particular service instance [21].

In order to minimize information exchange between repositories and agents, only a minimal set of information is replicated, i.e., only the information an agent needs to drive the execution of those process instances that it might potentially be responsible for (for which it provides services). Among all global OSIRIS services, the most important ones for distributed and decentralized process execution are the the process repository which holds the global definitions of all processes types, the service registry which is a directory of all available services in the system provided by OSIRIS agents, and the load repository which manages information on the load of all agents in the system.

OSIRIS' decentralized and distributed approach to process execution is illustrated in Figure 1. Different service types are depicted with different shapes, execution orders are illustrated by directed edges. Agents (OSIRIS layers) are sitting on top of all service providers and allow them to make available their services to OSIRIS processes. In the center, some of the core OSIRIS services are displayed. Figure 1 also shows how a process description is replicated in small pieces to the OSIRIS agents (dotted lines). Finally, after replication, enough process and service meta information is locally available to allow for peer-to-peer process execution. In particular, when a process is instantiated and executed, the OSIRIS agents can decide on their own, based on locally replicated information, where to route a request to (solid lines between OSIRIS agents). This makes sure that process execution takes place in a decentralized and distributed way and guarantees a high degree of scalability.

Distributed Concurrency Control for Processes. In databases, transactional execution guarantees are of high importance and have strong practical impact in a large number of applications. In general, this is also true for hy-

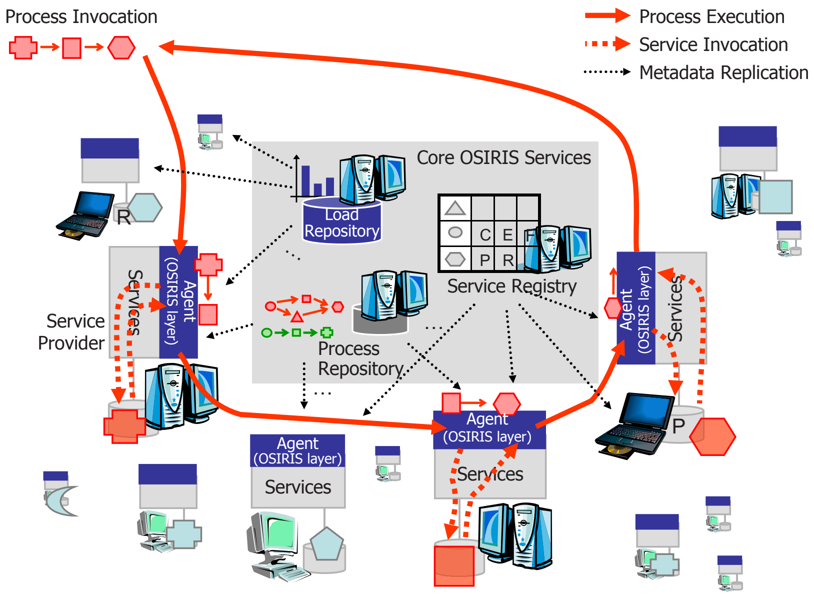


Fig. 1. Distributed Agent-based Execution of OSIRIS Processes

perdatabases. However, isolated and atomic behavior for concurrent distributed, service-based applications needs to take into account the higher level semantics of services (compared to rather low-level database operations). Transactional processes [19] consider these constraints and provide process support with transactional guarantees over distributed components using existing services as a generalization of traditional database transactions. Essentially, transactional processes exploit the termination semantics of the individual services they contain. Each service is either *compensatable*, *retriable*, or *pivot*, following the model of flexible transactions [27]. The effects of compensatable services can be semantically undone after the invocation has successfully returned. Retriable services are guaranteed to terminate correctly, even if they have to be invoked repeatedly. In this case, the last invocation succeeds while all previous invocations of this service do not leave any effects. [18] presents a more advanced distinction between termination classes, based on execution costs of services. Pivot services are those that cannot be compensated, due to the lack of an inverse service, or which are not appropriate for compensation due to their high costs.

On the basis of the transactional process model, the AMOR (Agents, MObility and tRansactions) approach [8,7] allows to provide global transactional guarantees, i.e., atomicity and isolation applied at the level of processes without any global component involved. Conventionally, isolation and atomicity are enforced using a locking protocol like the strict two-phase locking (2PL) in combination with a global commit protocol like the two-phase commit (2PC) [12] and require a centralized coordinator. These protocols are not applicable in completely distributed agent-based applications. AMOR uses a novel protocol which is based on decentralized serialization graph testing to ensure global correctness (concurrency control and recovery) in peer-to-peer environments without a global coordinator. Essentially, each agent is equipped with partial knowledge (local serialization graph containing information on conflicts with other agents) that allows them to coordinate. Globally correct execution is achieved by communication among dependent agents and can even be enforced in case of incomplete local knowledge. It can be guaranteed that each agent can decide at commit time whether it is able to safely commit its work or whether it has to wait on other agents to commit their work first before they can proceed.

Thus, AMOR provides the basic protocol that can be used to add transactional semantics to any kind of agent cooperation without imposing a dedicated infrastructure for this coordination, similarly to the way it has brought forward P2P systems with transactional semantics [3].

4 Conclusion

The significant growth of information over the last years has led to an increasingly large number of information sources in various formats and at different locations. This information might even be subject to frequent changes and complex interdependencies. In order to support applications in dealing with this wealth of heterogeneous information, novel approaches are required to access these information sources, to mediate between them and to provide value-added services. These problems are in the focus of information agents, which take over these tasks on behalf of their users or other agents. Most importantly, tasks are usually delegated to groups of (possibly specialized) agents which solve them in a collaborative way.

At the same time, this evolution has also led to a re-thinking of database research. Databases are no longer in the center of applications but are hidden to the applications behind service interfaces. Nevertheless, well known guarantees from databases like optimized access and transactional execution are still needed, but at a higher level of semantics, namely the invocation of services.

The activities of the information agent and database communities have led to highly complementary results and research activities are more and more directed towards bringing both fields closer together. So the rather rhetorical question from the title of this paper can be clearly answered: there is indeed a high potential for synergies and cross-fertilization between both fields which is visible in the promising results of some initiatives, and the research agendas will hopefully be much closer aligned in the near future.

In this paper, we have reported on some activities which originated from the database community and which, as we believe, will also have a strong impact also on cooperative information agents. In particular, we have summarized the hyperdatabase vision which aims at applying database system concepts outside of databases. In addition, with OSIRIS and AMOR, we have presented two hyperdatabase implementations. The first aims at providing optimized routing of service requests, as a generalization of query optimization in databases, while the latter focuses on providing transactional execution guarantees for composite services, as a generalization and extension of ACID guarantees known from database transactions.

Over the last ten years, the hyperdatabase vision has been implemented, exploited, and evaluated in a large variety of applications. However, the hyperdatabase vision is not static but needs to evolve with the ongoing advancements and new trends in large-scale, distributed and heterogeneous information spaces. Current activities in the context of the hyperdatabase vision, for instance, consider additional support for context-aware service composition and semantic failure handling. Essentially, when considering the current context (e.g., location) of a user or her individual preferences, personalized process-based applications can be either newly created or existing ones can be automatically adapted. This includes the customizaiton and generation of processes using semantic Web services, their reliable distributed execution, and finally the exploitation of semantics for failure handling purposes – the latter will significantly benefit from recent work done in the context of cooperative information agents.

References

1. Abiteboul, S., Agrawal, R., Bernstein, P.A., et al.: The Lowell Database Research Self-Assessment. Communications of the ACM 48(5), 111–118 (2005)
2. AgentLink. Special Interest Group on Intelligent Information Agents, http://www.dbgroup.unimo.it/IIA/

3. Antony, S., Agrawal, D., Abbadi, A.E.: P2P Systems with Transactional Semantics. In: Proceedings of the 11th International Conference on Extending Database Technology (EDBT 2008), Nantes, France, March 2008, pp. 4–15. ACM Press, New York (2008)
4. CIA. International Workshop Series on Cooperative Information Agents, http://www-ags.dfki.uni-sb.de/~klusch/IWS-CIA-home.html
5. Codd, E.F.: The Capabilities of Relational Database Management Systems. IBM Research Report, San Jose, California, RJ3132 (1981)
6. Gmach, D., Krompass, S., Scholz, A., Wimmer, M., Kemper, A.: Adaptive Quality of Service Management for Enterprise Services. ACM Transactions on the Web (TWEB) 2(1) (Febuary 2008)
7. Haller, K., Schuldt, H., Schek, H.-J.: Transactional Peer-to-Peer Information Processing: The AMOR Approach. In: Chen, M.-S., Chrysanthis, P.K., Sloman, M., Zaslavsky, A. (eds.) MDM 2003. LNCS, vol. 2574, pp. 356–361. Springer, Heidelberg (2003)
8. Haller, K., Schuldt, H., Türker, C.: Decentralized Coordination of Transactional Processes in Peer-to-Peer Environments. In: Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, pp. 28–35. ACM Press, New York (2005)
9. Klusch, M. (ed.): Intelligent Information Agents. Springer, Heidelberg (1999)
10. Lopes, A., Costa, P., Bergenti, F., Klusch, M., Blankenburg, B., Möller, T., Schuldt, H.: Context-aware Secure Service Composition Planning and Execution on E-Health Environments. In: Proceedings of the European Conference on eHealth (ECEH 2006), Fribourg, Switzerland, pp. 179–190 (October 2006)
11. Möller, T., Schuldt, H., Gerber, A., Klusch, M.: Next Generation Applications in Healthcare Digital Libraries using Semantic Service Composition and Coordination. Health Informatics Journal (HIJ), Special Issue on Health Digital Libraries 12, 107–119 (2006)
12. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 2nd edn. Prentice Hall, Englewood Cliffs (1999)
13. Papazoglou, M.P., Laufmann, S.C., Sellis, T.K.: An Organizational Framework for Cooperating Intelligent Information Systems. International Journal on Cooperative Information Systems 1(1), 169–202 (1992)
14. Pu, C., Schwan, K., Walpole, J.: Infosphere Project: System Support for Information Flow Applications. SIGMOD Record 30(1), 25–34 (2001)
15. Schek, H.-J., Böhm, K., Grabs, T., Röhm, U., Schuldt, H., Weber, R.: Hyperdatabases. In: Proceedings of the First International Conference on Web Information Systems Engineering (WISE 2000), Hong Kong, China, June 2000, pp. 14–25. IEEE Computer Society, Los Alamitos (2000)
16. Schek, H.-J., Schuldt, H.: The Hyperdatabase Project – From the Vision to Realizations. In: Proceedings of the 25th British National Conference on Databases (BNCOD 25), Cardiff, UK, July 2008. LNCS, vol. 5071. Springer, Heidelberg (2008)
17. Schek, H.-J., Schuldt, H., Weber, R.: Hyperdatabases: Infrastructure for the Information Space. In: Proceedings of the Sixth IFIP Working Conference on Visual Database Systems (VDB 2002), Bisbane, Australia, May 2002, pp. 1–15. Kluwer Academic Publishers, Dordrecht (2002)
18. Schuldt, H.: Process Locking: A Protocol based on Ordered Shared Locks for the Execution of Transactional Processes. In: Proceedings of the 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2001), Santa Barbara, CA, USA, May 2001, ACM Press, New York (2001)

19. Schuldt, H., Alonso, G., Beeri, C., Schek, H.-J.: Atomicity and Isolation for Transactional Processes. ACM Transactions of Database Systems (TODS) 27(1), 63–116 (2002)
20. Schuler, C., Schuldt, H., Türker, C., Weber, R., Schek, H.-J.: Peer-to-peer Execution of (Transactional) Processes. International Journal on Cooperative Information Systems 14(4), 377–406 (2005)
21. Schuler, C., Türker, C., Schek, H.-J., Weber, R., S.H.: Scalable Peer-to-Peer Process Management. International Journal of Business Process Integration and Management (IJBPIM) 1(2), 129–142 (2006)
22. Schuler, C., Weber, R., Schuldt, H., Schek, H.-J.: Peer-to-Peer Process Execution with OSIRIS. In: Orlowska, M.E., Weerawarana, S., Papazoglou, M.P., Yang, J. (eds.) ICSOC 2003. LNCS, vol. 2910, pp. 483–498. Springer, Heidelberg (2003)
23. Schuler, C., Weber, R., Schuldt, H., Schek, H.-J.: Scalable Peer-to-Peer Process Management – The OSIRIS Approach. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2004), San Diego, CA, USA, June 2004, pp. 26–34. IEEE Computer Society Press, Los Alamitos (2004)
24. Schumacher, M., Helin, H., Schuldt, H. (eds.): CASCOM: Intelligent Service Coordination in the Semantic Web. Whitestein (2008)
25. Silberschatz, A., Zdonik, S.B.: Database Systems - Breaking Out of the Box. SIGMOD Record 26(3), 36–50 (1997)
26. Wiederhold, G., Wegner, P., Ceri, S.: Toward Megaprogramming. Commununications of the ACM 35(11) (1992)
27. Zhang, A., Nodine, M.H., Bhargava, B.K.: Global Scheduling for Flexible Transactions in Heterogeneous Distributed Database Systems. IEEE Transactions on Knowledge and Data Engineering (TKDE) 13(3), 439–450 (2001)

Agents and Semantic Services: A Critical Review

Katia P. Sycara

School of Computer Science
Carnegie Mellon University
katia@cs.cmu.edu
http://www.cs.cmu.edu/~softagents

Abstract. *Web Services* have been hailed as the latest silver bullet for enabling business process representation and integration. In recent years, industry has developed a variety of standards, e.g. SOAP, WSDL, UDDI, BPEL for web services discovery, description, and distributed execution over the Web. These industry standards have emphasized description of service interfaces. However, they have many limitations with respect to flexible interaction and interoperability among heterogeneous services. For example, WSDL does not give any indication on the order of message exchange between a service and its client.

On the other hand, Multi-Agents Systems research over the years has developed techniques for autonomous and goal-directed agent interactions, agent communication languages that support extended conversations, flexible automated agent discovery in open environments, agent negotiation and methods for peer to peer reactive and proactive agent behaviors in dynamic environments.

In this talk, I will present requirements and extensions on web services functionality for supporting business processes. Some of these extensions include peer to peer and multi-party interactions, dynamic on the fly-composition of web services, message patterns that go beyond request-response, contracts and service level agreements. In addition, I will present characteristics of agents and web services that encourage fruitful application of techniques from agents to services and vice versa. In particular, I will articulate the importance of formally specified, unambiguous semantics for increasing service interoperability and flexibility of interactions, thus bringing the services and agents paradigms and technologies closer to one another. A first step towards this rapprochement is the development of formal languages and inference mechanisms for representing and reasoning with core concepts of Web Services.

In closing, I present my vision of Web services as *autonomous goal-directed agents* which select other agents to interact with, and flexibly negotiate their interaction model, acting in peer to peer fashion. The resulting Web services, that I call *Autonomous Semantic Web services*, utilize ontologies and semantically annotated Web pages to automate the fulfillment of tasks and transactions with other Web agents. In cross-fertilizing each other, both agent technology and web services technology can discover new synergies that will make the combination and subsequent adoption much stronger, vital and useful than either of the two technologies separately.

Agent-Supported Planning in Distributed Command and Control Environments

James H. Lawton

US Air Force Research Laboratory
Information Directorate, Rome Research Site
Rome, NY, 13441, USA
James.Lawton@rl.af.mil

Abstract. To be able to meet the future challenge of employing forces anywhere in the world in support of national security objectives, modern military forces require highly synchronized, distributed planning and re-planning capabilities that are sufficiently flexible to adapt to any level of conflict. This talk will present a research program underway at the USAF Research Laboratory's Information Directorate known as DEEP (Distributed Episodic Exploratory Planning). DEEP is an agent-based distributed planning system that has been designed to support future military command and control (C2) operations. The talk will discuss the motivation for moving from a centralized planning model to a distributed mixed-initiative approach, along with the DEEP architecture and the key research challenges for achieving this vision. The distributed agent-supported planning capabilities, which utilize past experience to solve current problems, will be emphasized[1].

1 Introduction

The U.S. and other highly industrialized nations have developed military capabilities that excel in conventional force-on-force warfare, especially where tactics are well developed and known. However, modern adversaries have devised the strategy of not going head-to-head with these capabilities and instead combat modern conventional forces with unconventional tactics. One example of the result of a weapon system being vastly superior is the case of the air superiority fighter which modern adversaries totally avoid putting themselves in a position to contest them.

To meet these future challenges, U.S. forces are in the midst of a transformation to not only support traditional high-tempo, large force-on-force engagements, but also smaller-scale conflicts characterized by insurgency tactics and time-sensitive targets of opportunity. This transformation requires a vastly new Command and Control (C2) process that can adapt to the any level of conflict, provides a full-spectrum joint warfighting capability, and can rapidly handle any level of complexity and uncertainty.

[1] For a more complete description of the DEEP project, see [AFRL-08a], [AFRL-08b] and [AFRL-08c].

To meet future challenges, the U.S. Air Force (USAF) is moving towards a model of continuous air operations not bounded by the traditional 24-hour Air Tasking Order (ATO) cycle. Meeting these objectives will require a highly synchronized, distributed planning and replanning capability. As a potential way ahead, in May 2006 the USAF released a revolutionary vision paper [Braun-06] depicting what a potential future C2 environment could be. Four key concepts emerged as being critical to the success of this vision of a future Air Operations Center (AOC):

– Distributed/Reachback planning
– Redundant/Backup planning
– Continuous planning
– Flexible, scalable, tailorable C2

Experience with recent operations also reveals that the C2 process must transition from a process of observation and reaction to one of prediction and preemption. To achieve this, we will need to go beyond the focus of military operations, and instead address the entire spectrum of Political, Military, Economics, Social, Infrastructure, and Information (PMESII).

To that end, the focus of the research discussed in this talk has focused on developing a C2 environment that supports the vision of Network Centric Operations (NCO). The tenets of NCO are:

– Information sharing
– Shared situational awareness
– Knowledge of commanders intent.

2 DEEP: Distributed Episodic Exploratory Planning

In response to the need to support these key NCO tenets, the long-term goal of the Distributed Episodic Exploratory Planning (DEEP) project is to develop in-house a prototype system for distributed, mixed-initiative planning that improves decision-making by applying analogical reasoning over an experience base. The two key objectives of DEEP are:

– Provide a mixed-initiative planning environment where human expertise is captured and developed, then adapted and provided by a machine to augment human intuition and creativity.
– Support distributed planners in multiple cooperating command centers to conduct distributed and collaborative planning.

That is, the architecture of DEEP was explicitly designed to support the key tenets of NCO in a true distributed manner. Because DEEP is not based on any current C2 system, we are able to explore concepts such as combining planning and execution to support dynamic replanning, to examine machine-mediated self-synchronization of distributed planners, and to experiment with the impact of trust in an NCO environment (e.g., Good ideas are more important than their source).

Alberts and Hayes [Alberts-07] advocate bold new approaches beyond current organizational process, focusing on what is possible for NCO. High priority basic research topics recommended as areas to systematically explore are:

1. Taxonomy for planning and plans
2. Quality metrics for planning and plans
3. Factors that influence planning quality
4. Factors that influence plan quality
5. Impact of planning and plan quality on operations;
6. Methods and tools for planning
7. Plan visualization

This talk describes our approach to achieving this vision of NCO by presenting the progress to date on the development of the DEEP prototype, especially as it relates to these priorities.

2.1 High Level Architecture

The DEEP architecture, shown in Figure 1, is a systems-of-systems design comprised of the following sub-systems:

– Distributed Blackboard for multi-agent, non-deterministic, opportunistic reasoning
– Case-Based Reasoning system to capture experiences (successes and/or failures)
– Episodic Memory for powerful analogical reasoning
– Multi-Agent System for mixed initiative planning
– ARPI Core Plan Representation [Pease-98] for human-to-machine common dialog
– Constructive Simulation for exploration of plausible future states

The key components of the DEEP architecture include a messaging system, various knowledge objects, a shared data storage system, and a variety of agents for manipulating plans. For convenience, we will describe the pieces in the architecture in the order in which might be typically used. One should bear in mind, however, that in this type of mixed-initiative system, there will rarely be a clean path from the initial planning problem to the final solution.

Consider the system-of-systems given in Figure 1. The starting point for entry into the system occurs when a commander describes a new mission using a planning agent (1). The planning agent allows a commander to input information into the system which defines their current objectives. These objectives, along with other information, such as resources, locations, and time constraints, are collectively known as the *situation*. This situation is then placed on the shared blackboard (2). The blackboard would in turn notify all registered components of the existence of the new situation. The other planning agents, with their

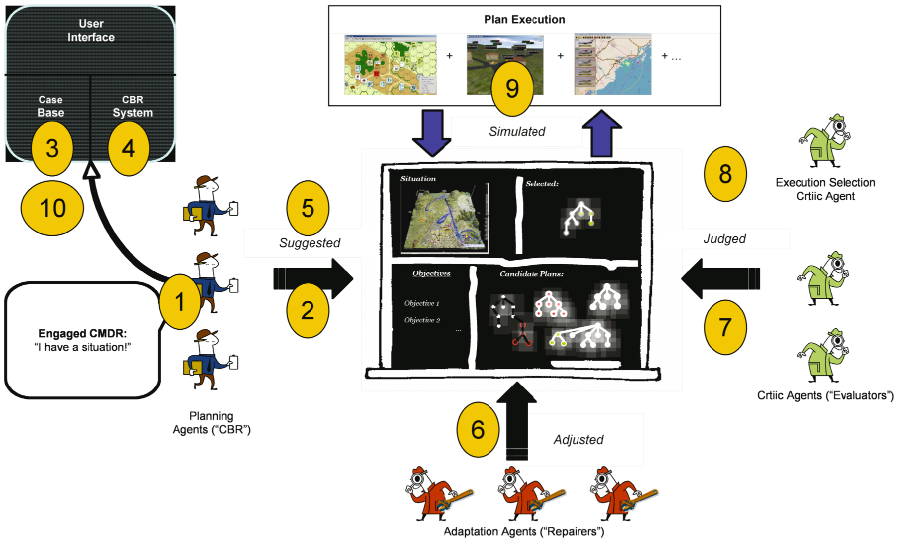


Fig. 1. DEEP High Level Architecture

associated case bases and cased-based reasoning capabilities, each search their case base using the situation given for relevant past experiences (3). These results are then modified to fit the current situation (4) and are posted to the blackboard as candidate plans (5). Once the candidate plans are on the blackboard, they are adapted by specialized adaptation agents to further refine these plans to meet the current situation (6). These plans are now ready to be critiqued by the critic agents. These agents concurrently scrutinize the candidate plans and score them based on their individual expertise (7). Once the plans are scored, the execution selection critic gathers the adapted plans along with their scores, determines their overall scores, and selects a number of top rated plans to be executed (8). The top rated plans are now executed (currently in a simulated environment) (9). Once a plan completes execution, the results are combined with the plan and assimilated back into the original planning agent's case base (10).

Although we have described this planning and execution as a single flow through the system, in reality few plans will execute without changes. The DEEP architecture supports the modification of currently executing plans through feedback of partial results of plan execution into the blackboard. This allows the plans to be run through the adaptation and critique processes as many times as needed (10).

References

[Alberts-07] Alberts, D., Hayes, R.: Planning: Complex Endeavors. Command and Control Research Program Press (2007)

[AFRL-08a] DeStefano, C., Lachevet, K.K., Carozzoni, J.A.: Distributed Planning in a Mixed-Initiative Environment. In: Proceedings of the 13th International Command and Control Research and Technology Symposium (2008)

[AFRL-08b] Staskevich, G.R., Lawton, J.H., Carozzoni, J.A.: Semantic Interoperability in Distributed Planning. In: Proceedings of the 13th International Command and Control Research and Technology Symposium (2008)

[AFRL-08c] Ford, A.J., Lawton, J.H.: Synthesizing Disparate Experiences in Episodic Planning. In: Proceedings of the 13th International Command and Control Research and Technology Symposium (2008)

[Braun-06] Braun, G.: AFFOR Command and Control Enabling Concept–Change 2. AF/A5XS Internal Report (May 25, 2006)

[Pease-98] Pease, A.: Core Plan Representation. Version 4 (November 6, 1998)

Towards Trust-Based Acquisition of Unverifiable Information

Eugen Staab, Volker Fusenig, and Thomas Engel

Faculté des Sciences, de la Technologie et de la Communication,
Université du Luxembourg,
Campus Kirchberg, 6, rue R. Coudenhove-Kalergi, L-1359 Luxembourg
{eugen.staab,volker.fusenig,thomas.engel}@uni.lu

Abstract. We present a trust-based mechanism for the acquisition of information from possibly unreliable sources. Our mechanism addresses the case where the acquired information cannot be verified. The idea is to intersperse questions ("challenges") for which the correct answers are known. By evaluating the answers to these challenges, probabilistic conclusions about the correctness of the unverifiable information can be drawn. Less challenges need to be used if an information provider has shown to be trustworthy. This work focuses on three major issues of such a mechanism. First, how to estimate the correctness of the unverifiable information. Second, how to determine an optimal number of challenges. And finally, how to establish trust and use it to reduce the number of challenges. Our approach can resist collusion and shows great promise for various application areas such as *distributed computing* or *peer-to-peer networks*.

Keywords: Information acquisition, trust.

1 Introduction

A lot of research addresses trust that is based on direct experiences [1,2]. These direct experiences result from evaluating the outcomes of interactions with other agents. Such an evaluation however is not possible when the outcome of an interaction is information that cannot be verified, or the verification would be too costly. An example illustrates this situation:

Example 1. Agent Alice wants to know the result of $15 + 8$. However, Alice cannot compute the result because she is out of resources at the time. So Alice asks another agent Bob to do it for her. Although Bob knows how to calculate the result, he returns the wrong result 26 because he is malicious and wants to harm Alice. Consequently, Alice, who does not want to verify the result because she wanted to save resources, uses the wrong result in her further work. This will cause additional costs for her, and if she is not aware of them, she even does not classify the experience with Bob as a negative experience.

An attempt to solve this problem is to ask several agents for the desired information, to compare their answers, and to discard these if they are not the same (see e.g. [3]). However, this approach is sensitive to *collusion* [2], especially in settings with only few information providers, and its efficiency ought to be improved. Therefore, we propose an alternative approach. The main idea is to merge requests for information with so called "challenges" for which the correct answers are already known. The requesting agent evaluates the responses to the challenges and draws conclusions about the responses to the "real" requests. This leads to an estimation of another agent's trustworthiness which in turn can be used to reduce the number of challenges that need to be used. However, a minimal number of challenges is always retained to account for the *first-time offender problem* [4].

The remainder of the paper is organized as follows. In Section 2, we outline several application scenarios for the mechanism and show how challenges can be generated in each scenario. The mechanism for information acquisition is presented in Section 3. We discuss several issues concerning the practical use of the mechanism in Section 4. Section 5 is used to refer to related work. We conclude our paper and give an outlook on future work in Section 6.

2 Application Scenarios

This section outlines some application areas for which the mechanism shows great promise.

The mechanism can be used in cases where calculations are outsourced and the results shall not be verified. As it was motivated in Ex. 1, our mechanism can be applied to the scenario of *distributed computing*, more specifically *grid-computing* [5] or *cloud-computing* [6]. In these cases, challenges can either be provided by trusted nodes or be computed whenever the system of the requesting agent is idle.

Another application scenario is the exchange of *routing information* in *Wireless Ad Hoc Networks* [7]. As new routing information cannot be verified, the trust-based mechanism would help to enforce the provision of reliable information. The challenges can be chosen to be questions about routes that are known to exist (e.g. because packets have been sent over these routes in the recent past).

In *peer-to-peer networks*, our trust-based mechanism can be used against pollution and poisoning attacks (see [8]). Challenges would consist of requests for files that already have been verified by a human to match their description and to be free of "bad chunks". Note that in these settings, a small number of challenges for a given number of real requests would be essential for the practicability of the mechanism. Also, the verification of a partly downloaded response to the challenge should start as soon as a certain amount of packets is received.

The mechanism can also be useful for the purpose of exchanging *reputation information* such that it implements the concept of "semantic distance" which was introduced by Abdul-Rahman and Hailes [9]. The answers to challenges would be used to determine the semantic distance, which in turn could be used to weight the answers to the real requests. Related to that, the trust-based mechanism can be used in *Multi-Agent Systems*, in which *beliefs* about the world (which cannot easily be verified) are exchanged. The mechanism would prevent the acquisition and the spread of wrong or outdated beliefs as well as it would filter out "incompatible" beliefs. In this case, challenges would be based on the *knowledge* of an agent.

3 Trust-Based Mechanism for Information Acquisition

In the following, we describe one run of the mechanism. An agent wants to get answers to m questions. We will call these questions "real requests". In our particular case, the agent will not be able to verify the correctness of the answers which he will get to the real requests (because, as mentioned earlier, he is incapable or does not want to spend resources on it). Therefore, before sending the request, the agent adds n challenges for which the answers are known to him. These challenges must be chosen in a way such that another agent is not able to easily distinguish them from real requests; how this choice can be made depends on the concrete setting (see Sect. 2 for examples). The agent merges the m requests and the n challenges into a vector of size $m + n$, in an equally distributed manner. This *request-vector* is then transferred to the information provider which is expected to reply with a *response-vector* of the same size. After reception, the response-vector is evaluated, i.e. the answers to the challenges are verified. The resulting error rate is used for two things. First, to estimate the error rate of the answers to the real requests (see Sect. 3.1) and second, as input for a trust-update algorithm (see Sect. 3.2). It is shown how an optimal number of challenges can be computed when an information provider is not known (see Sect. 3.3) and how the established trust can be used to reduce this number of (costly) challenges (see Sect. 3.4). Finally, a decision needs to be made whether the obtained response-vector is accurate enough or a new request to other information providers should be done. This decision-making process however is not part of this work (see Sect. 4 for a discussion).

3.1 Evaluation of a Response

An agent A sends a request-vector with $m + n$ questions to an information provider who is expected to reply with a response-vector of the same size. Agent A evaluates the answers to the n challenges in such a response-vector, and finds r correct and s incorrect answers, with $r + s = n$. We will call the rate of incorrect answers $\frac{s}{s+r}$ the *error rate*. As the challenges and real requests in the request-vector were equally distributed, r and s can be used to estimate a probability distribution for the error rate in the remaining part of the response-vector (the part with the *real requests*). In this work we will assume that the information provider can be described by an error-probability p_w: with probability p_w he answers independently each question incorrectly (see Sect. 4 for a discussion of this assumption). Let $\{x_i\}_{i=1}^n$ denote the evaluated answers to the challenges, so each x_i is in $\{correct, incorrect\}$. We can use *Bayes' theorem* to get the probability of each error-probability p_w:

$$P(p_w|\{x_i\}_{i=1}^n) = \frac{P(\{x_i\}_{i=1}^n|p_w)P(p_w)}{P(\{x_i\}_{i=1}^n)} \qquad (1)$$

We can simplify this formula in the following way. First, the denominator can be calculated by marginalization over all rates that could have produced the given observations. Second, no prior information on p_w is given, so we have to assume all p_w to be equally probable, and so it can be left out (it's probability density function is 1 in $[0, 1]$). Third, from statistical independence between all single answers it follows that $P(\{x_i\}_{i=1}^n | p_w) = p_w^s (1 - p_w)^r$. As a result, we get the probability density function of the *beta distribution* with parameters $\alpha = s + 1$ and $\beta = r + 1$:

$$f(p_w; s+1, r+1) = \frac{p_w^s (1 - p_w)^r}{\int_0^1 x^s (1 - x)^r dx} \tag{2}$$

Note that this probability distribution estimates the error probability p_w of the information provider and hence estimates also the error rate of the answers to the real requests.

The mean of the beta distribution is given by $\frac{\alpha}{\alpha+\beta}$ [10]. So, having costs c_w for one wrong answer, the expected costs for using the answers to the real requests can be computed as follows:

$$\begin{aligned} E[m * c_w * f(p_w; s+1, r+1)] &= m * c_w * E(f(p_w; s+1, r+1)) \qquad (3) \\ &= \frac{m * c_w * (s+1)}{s+r+2}. \qquad (4) \end{aligned}$$

3.2 Bayesian Trust-Model

In this section, we present a formal trust-model which represents an agent's trust in another agent by a *trust-value* t and the corresponding *uncertainty* u. These values will be used in Sect. 3.4 to reduce the number of challenges that are needed for a request. We describe and justify how t and u are computed and define initial values.

Let (t_{AB}, u_{AB}) denote the trust that an agent A has in an information providing agent B. In our model, a trust-value $t_{AB} \in (0, 1)$ is A's estimation of the rate of correct answers in a response-vector that will eventually be received from B. The corresponding uncertainty $u_{AB} \in (0, 1]$ describes how certain A is about this trustworthiness estimation t_{AB}. Having no experience with B, an agent A trusts always with $(t_{AB}, u_{AB}) = (0.5, 1)$, i.e. the initially expected error probability is 0.5 but this is believed with the highest possible uncertainty. This seems intuitive but also results from the formulas (which are defined below) when no information is given.

Several forms of representation for trust have been proposed in literature (for an overview see [11]). While uncertainty is usually represented as in our case, there are several different representations of trust-values. To us, it seems to be a mere issue of convention. However we want to justify our choice for $(0, 1)$ because, based on experiences, full trust ($t = 1$) or distrust ($t = 0$) seem not to be reasonable – even though full trustworthiness and untrustworthiness are possible.

Trust-Values. Let r^B_{old} and s^B_{old} denote the amount of respectively correct and incorrect answers from past response-vectors received from an agent B. More precisely, to give recent experiences a higher importance than older ones, each single experience in r^B_{old} and s^B_{old} has been weighted with a so called "aging factor" λ (q.v. [11]). The trust-value t_{AB} is defined to be the expected value of a beta distributed random variable X with parameters $\alpha = s^B_{old} + 1$ and $\beta = r^B_{old} + 1$:

$$t_{AB} \overset{def}{=} E(X) = \frac{r^B_{old} + 1}{r^B_{old} + s^B_{old} + 2} \tag{5}$$

The probabilistic justification for this calculation follows directly from Section 3.1: the trust-value is the most probable error probability p_w based on past observations. Formula (5) constitutes the core part of the *trust-value* computation in the proposed mechanism. We want to emphasize the possibility to extend this computation with other approaches for trust-value computation that have been proposed numerously in literature (e.g. see [1,2]).

Uncertainty. The uncertainty u_{AB} in our model is based on the *variance* σ^2 of the beta distribution, i.e. the lower the expected variance, the lower the uncertainty. We will show that for parameters $\alpha \geq 1$ and $\beta \geq 1$, the variance of the beta distribution has the two properties that Wang and Singh [12] claim for certainty measures. However, in comparison to their approach and the one used in TRAVOS [13] that both compute integrals over a beta distribution, the variance of the beta distribution is much easier to compute. For a beta distributed random variable X with parameters α and β, the variance σ^2 is given by ([10]):

$$\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{6}$$

As parameters for σ^2 we have again $\alpha = r^B_{old} + 1$ and $\beta = s^B_{old} + 1$, so $\alpha, \beta \geq 1$ because $r^B_{old}, s^B_{old} \geq 0$. The highest value the variance can take in this case is $1/12$ at point $\alpha = \beta = 1$ (which will follow directly from Theorems 1 and 2). We normalize σ^2 accordingly and get u_{AB}:

$$u_{AB} \overset{def}{=} 12\sigma^2 \tag{7}$$

It remains to show that the variance has the properties demanded by Wang and Singh [12] – or rather the *inverse* properties, because we are addressing uncertainty (they addressed certainty). First, we show that for fixed *conflict* χ between α and β, σ^2 decreases. Conflict refers to the similarity of α and β: Many positive *and* negative experiences give reason for a higher uncertainty about an agents trustworthiness. Without loss of generality we assume $\alpha \leq \beta$ such that the conflict can be expressed as $\frac{\alpha}{\beta}$. Second, we show that for increasing conflict and increasing α, σ^2 decreases to $\frac{\alpha+\beta}{2}$.

Theorem 1. *For $\alpha \leq \beta$ and fixed conflict $\chi := \frac{\alpha}{\beta}$, σ^2 decreases for increasing $\alpha + \beta$.*

Proof. We have to show that for any $\chi > 0$ the first derivative of σ^2 is negative for all possible α, β. In (6) we substitute β by (α/χ) and differentiate with respect to α:

$$\frac{d\sigma^2}{d\alpha} = \frac{d}{d\alpha}\left(\frac{\alpha^2/\chi}{(\alpha/\chi+\alpha)^2(\alpha/\chi+\alpha+1)}\right) \tag{8}$$

$$= \cdots = -\frac{\chi^2}{(1+\chi)(\chi+\alpha+\chi\alpha)^2} < 0, \forall \beta \geq \alpha \geq 1. \tag{9}$$

□

Theorem 2. *Assuming fixed $\gamma := \alpha + \beta$. For increasing conflict, σ^2 increases.*

Proof. We have to look at the "slices" of σ^2 where $\gamma := \alpha + \beta$ is fixed. The function at the respective "slice" should increase when the conflict increases and decrease again when the conflict decreases. In σ^2 we substitute β by $(\gamma - \alpha)$:

$$g(\alpha) := \frac{\alpha(\gamma-\alpha)}{\gamma^2 * (\gamma+1))} \tag{10}$$

The maximum of $g(\alpha)$ is to be shown to be at the point where the conflict is maximal, i.e. $\alpha = \beta$ which is $\alpha = \gamma/2$. If additionally $g(\alpha)$ is concave down, i.e. the second derivation is negative for all possible α and γ, we're done. We find:

$$\frac{d^2 g(\alpha)}{d\alpha^2} = \cdots = \frac{dg(\alpha)}{d\alpha}\left(\frac{\gamma-2\alpha}{\gamma^2+\gamma^3}\right) = -\frac{2}{\gamma^2+\gamma^3} \tag{11}$$

For $\gamma > 0$ the first derivation has its only root (and so its maximum) clearly at $\alpha = \gamma/2$. The second derivation is negative for all $\gamma > 0$. □

Depending on the scenario, it might be necessary to control the speed with which the uncertainty decreases; however this is done (e.g. taking the nth root of the variance), it has to be guaranteed that the properties of the variance described in Theorems 1 and 2 are still fulfilled.

3.3 Optimizing the Number of Challenges

In this section, we show how to find an optimal number of challenges n when given a number of real requests m. With "optimal" we refer to a number of challenges that minimizes the expected costs that arise when the error rate for the challenges and the one for the real requests differs. For example, an agent gets many correct answers to the verifiable challenges but not a single correct answer to the real requests. Then, the agent would underestimate the error rate for the real requests and work with incorrect information. Therefore, we find the number of challenges for which the expected difference between errors to the challenges and errors to the real requests is minimized. Note that a malicious information provider could try to answer all challenges correctly while giving wrong answers to all real requests. However, such situation could only be reached by *guessing* the number and the positions of the challenges – because real requests and challenges are randomly merged. We proceed as follows. We first calculate the probabilities for all possible differences in the error rates. These probabilities are used to determine the expected costs. The resulting cost-function is minimized in respect to n.

First, let us virtually separate challenges from real requests and denote the vector that contains the answers to the challenges with $\vec{n}$ and accordingly the vector that contains the answers to the real requests with $\vec{m}$. Let $\vec{m}$ be of size m and $\vec{n}$ be of size n. Further let $w(\vec{x})$ be a function returning the error rate in some vector $\vec{x}$. What we want to know first is the probability of having an error rate j in $\vec{m}$ given an error rate i in $\vec{n}$:

$$P\left(w(\vec{m}) = j | w(\vec{n}) = i\right) \tag{12}$$

The error rates $w(\vec{m})$ and $w(\vec{n})$ seem to be statistically independent. That is however not the case because they both depend on the same error probability p_w, according to which the answers were answered incorrectly (see also Sect. 3.1). Therefore, we have to consider $P(\vec{m}, p_w|\vec{n})$ (for the moment ignore the function $w(\cdot)$). We use the basic product rule (see [14], p. 51) and get:

$$\begin{aligned} P(\vec{m}, p_w|\vec{n}) &= P(\vec{m}|p_w, \vec{n})P(p_w|\vec{n}) & (13)\\ &= P(\vec{m}|p_w)P(p_w|\vec{n}) & (14)\end{aligned}$$

The last step leading to (14) is allowed because $\vec{m}$ is independent of $\vec{n}$ for given p_w. The probability $P(w(\vec{m}) = j|p_w)$ is the probability for k failures in m independent Bernoulli trials with error probability p_w, where $k = j * m$. So, we have a binomial distribution with parameters m and p_w; we will write $P_{p_w}(k|m)$. The probability $P(p_w|w(\vec{n}) = i)$ follows the beta distribution f with parameters $\alpha = i * n + 1$ and $\beta = (1 - i) * n + 1$ (as derived in Section 3.1). In order to get (12) we can integrate over all mutually exclusive p_w:

$$\begin{aligned} P(w(\vec{m}) = j|w(\vec{n}) = i) = \int_0^1 & P_{p_w}(j * m|m) & (15)\\ & * f(p_w; i * n + 1, (1 - i) * n + 1)dp_w & (16)\end{aligned}$$

Now, we can calculate the probability that the error rate in $\vec{m}$ differs from the error rate in $\vec{n}$ by some x. This is done by marginalization over all $\tau \in T$, where T is the set that contains all possible error rates in $\vec{n}$, i.e. $T = \{\frac{a}{n}|a \in \mathbb{N}_0, a \leq n\}$:

$$\begin{aligned} P(w(\vec{m}) - w(\vec{n}) = x) &= \sum_{\tau \in T} P(w(\vec{m}) = \tau + x, w(\vec{n}) = \tau) & (17)\\ &= \sum_{\tau \in T} P(w(\vec{m}) = \tau + x|w(\vec{n}) = \tau)P(w(\vec{n}) = \tau) & (18)\end{aligned}$$

Note that a priori all $\vec{n}$ are equiprobable and so we can compute $P(w(\vec{n}))$ in a combinatorial fashion; i.e. for $w(\vec{n}) = a/n$, we have to divide the number of possibilities to have a incorrect answers in $\vec{n}$, by the number of all possible vectors $\vec{n}$:

$$P(w(\vec{n}) = a/n) = \frac{\binom{n}{a}}{2^n} \tag{19}$$

Formula (18) will be calculated for all those differences x that are "possible", i.e. those contained in the set $\mathbb{X} := \{\frac{a}{m} - \frac{b}{n} | a, b \in \mathbb{N}_0, a \leq m, b \leq n\}$. Note that $\mathbb{X}$ contains both positive and negative x. For positive x, we have a higher error rate in $\vec{m}$, for negative x we have a higher error rate in $\vec{n}$.

We are now ready to define a cost-function that calculates the expected costs for a specific number of challenges n. Let the following parameters be given:

m – number of real requests,
c_c – costs for generating one challenge + costs for requesting the answer (the *information provider* may get some payment) + costs for evaluating the answer (which is a simple comparison to the already known answer),
c_d – costs for a difference between $w(\vec{n})$ and $w(\vec{m})$ of 1. So, a high c_d aims at a high accuracy in the estimation of $w(\vec{m})$.

Then, for a chosen number of challenges $n \geq 1$ the cost function is given by:

$$c(m, n, c_c, c_d) = n * c_c + \sum_{x \in \mathbb{X}} |x| * c_d * P(w(\vec{m}) - w(\vec{n}) = x) \qquad (20)$$

This cost function (20) adds the costs for using n challenges, to the expected costs when using n challenges for m requests. To find an optimal n, the function (20) needs to be minimized in respect to n. As an optimization of this cost function at runtime would be a too costly computation, we propose to use precomputed values for various m, c_c and c_d. In (20) we can take c_d out of the sum and rewrite the cost function as

$$c(m, n, c_c, c_d) = n * c_c + c_d * d(m, n) , \qquad (21)$$

where $d(m, n)$ stands for the remaining part in (20). This allows for computing $d(m, n)$ before actually running the mechanism and minimizing $c(\cdot)$ once c_c and c_d are determined.

Figure 1 shows $d(m, n)$ for specific m and n. Note that interestingly for fixed m and increasing n, the function $d(m, n)$ does not necessarily decrease. For example for $m = 7$, the expected difference is smaller for $n = 7$ than for $n = 8$ and smaller for $n = 14$ than for $n = 15$. The reason is that for $n = 7$, each error rate in $\vec{n}$ has an exact matching error rate in $\vec{m}$; but for $n = 8$, most error rates in $\vec{n}$ differ from all error rates in $\vec{m}$ what increases the expected difference of error rates. Apparently, this fact has an impact on $d(m, n)$ in all cases where n is a multiple of m or where n divides m (e.g. see $m = 10$ and $n = 5$).

3.4 Weighting the Number of Challenges with Trust

Up to now, we estimated the optimal number of challenges for a given number of real requests. For this calculation, we assumed that the information providing agent has not yet shown to be trustworthy. Looking at the case in which an agent would fully trust in an information provider, he would not need to use challenges any more. This justifies the consideration that the more trustworthy the opponent

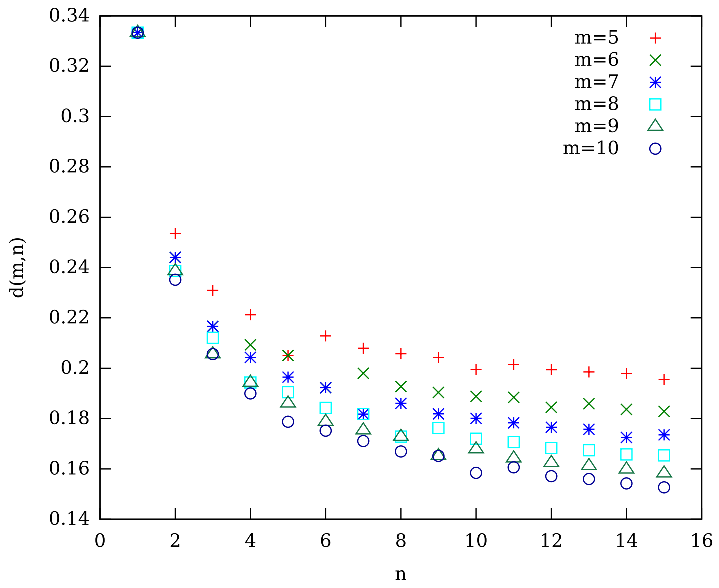


Fig. 1. Expected differences $d(m, n)$ between the error rates in $\vec{m}$ and $\vec{n}$

has shown to be, the smaller the number of challenges can be. In the following, we will integrate this notion of trust-based information acquisition into our model.

Basically, we want to weight the number of challenges that is optimal for the general case with the trust A has in the actual information provider B. As introduced in Section 3.2, A's trust in B is represented by a tuple $(t_{AB}, u_{AB}) \in (0, 1) \times (0, 1]$ with t_{AB} being the trust-value and u_{AB} the attached uncertainty (for simplicity we write t and u instead of t_{AB} and u_{AB} respectively). In this paper, we exclusively used probability theory to compute trust, as this can be formally justified. At this point however it seems appropriate to make use of our intuition and try to find the simplest formula that matches this intuition. The higher the trust-value t is, the smaller the number of challenges should be; the higher the attached uncertainty u is, the smaller the impact should be that the trust-value has on the number of challenges. This is met by $(1 - (1 - u)t)$, because for low trust-values or high uncertainty we take the full number of challenges ($\lim_{t\downarrow 0} 1 - t + tu = 1$ and we get 1 for $u = 1$), for low uncertainty the trust-value weights ($\lim_{u\downarrow 0} 1 - t + tu = 1 - t$) and for high trust-values the uncertainty weights ($\lim_{t\uparrow 1} 1 - t + tu = u$). So we get a number of challenges n that is weighted with trust by:

$$n = (1 - t + tu) \operatorname*{argmin}_{n'} c(m, n', c_c, c_d) \tag{22}$$

As desired, for initial trust $(t, u) = (0.5, 1)$ (see Sect. 3.2), the number of challenges is not reduced.

To account for changes in an agent's character or strategy and to prevent the unawareness towards the *first-time offender problem* [4], the number of challenges should never become zero. For that purpose, we introduce a minimum number of challenges $min_n > 0$. Finally, an agent determines the number of challenges he will use by $\max(n, min_n)$.

4 Discussion

Several aspects for the use of the presented mechanism in practice have not yet been addressed and shall be discussed in this section.

Collusion. For the choice of challenges, two important rules have to be respected in order to avoid the possibility of *collusion*:

1. For requests that were not answered satisfactory and are therefore requested again from other agents, the same challenges are to be used.
2. For requests for differing information, different challenges are to be used.

The reader can easily verify that otherwise colluding agents would be able to identify real requests and challenges only by comparing the different request-vectors and checking what has changed.

Distributing questions over time. Depending on the setting, it might be impractical to send requests together with challenges bundled in a vector. If an agent wants to send only one question at a time, he can distribute challenges and real requests over time. In this case, he has to be careful to give an attacker no opportunity to deduce information from the points in time when questions are sent about whether a question is a challenge or a real request (similar attacks are called *timing attack* in cryptology).

Lack of resources. In specific cases where a lack of resources is the only reason for not being able to verify a whole response, real requests can be declared to be challenges after a response has been received. This has the advantage that challenges cannot be disclosed (there are no challenges beforehand) and do not cause additional costs. Then, an optimal number of challenges can be determined *during* verification by using statistical considerations. Analogous problems can be found in the area of *Statistical Quality Control* ([15]), where a fraction of the output of a system is selected randomly and tested.

Delayed verification. Assume the case where the verification of information is impossible because some knowledge is not given that would be needed for the verification. To give an example, let agent A have requested the circumference c of a circle given its diameter d (with $c = \pi * d$). Let us also assume that A did not know π at request-time and could not verify the response. When A obtains π at a later point in time, he can verify the response by hindsight. The same holds, if A did not verify the answer because he was too busy at the time but gets round to verify the data at a later point in time. In our mechanism, such delayed verifications can be easily integrated by just recalculating the trust in the information provider and rechecking the need for further requests.

Context-sensitivity. The *context-sensitivity* of trust is important in the field of information acquisition. Agents may be competent in some domains ("what is the prime factorization of 12345?") and incompetent in others ("will it rain today in Prague?"). In order to account for the assumption that an information provider can be described by an error-probability p_w (see Sect. 3.1), all questions in one request vector must belong to the same domain. Still, different request-vectors can belong to different domains. The approach by Rehák and Pechoucek [4] seems suitable to account for the latter issue. Alternatively, techniques such as *Latent Semantic Indexing (LSI)* [16], *PLSI* [17] or *Concept Indexing* [18] can be used. These techniques would allow for defining the context-space on the basis of acquired natural-language text.

Selection of information providers. How to decide whether a response-vector is accurate enough was not part of this work. However, the general idea is that an agent should discard acquired information and request it from other agents if he can expect that the improvement will be worth the investment. In case he decides to discard the information and request it again, he has to choose an information provider from which to request. Solutions to this problem, known as the *exploitation vs. exploration* problem, can be found in literature [19,20].

Efficiency. In the approach of redundant computation (see Sect. 1), the requesting agent needs to choose a positive integer $\rho > 1$ as redundancy level (usually $\rho = 2$ or $\rho = 3$ [3]). Opposed to this, in our approach, the requesting agent can adjust continuously the amount of resources used for verification. To give an example, if the ratio of challenges n to real requests m is chosen to be $2/4$, then this would amount to a redundancy level of $\rho = 0.5$ if that was possible.

However, the generation of challenges has also to be taken into account. As this generation is domain dependent, we will analyze an approach that is generally possible[1]: the results from redundant requests are used to fill a pool that contains the answers for future challenges; as soon as the pool contains n answers, these are used for a request-vector of size $m+n$ with m real requests and n challenges. A mechanism that only uses redundancy gives n results for $\rho * n$ requests, i.e. an efficiency of $\frac{1}{\rho}$. By additionally using challenges from the pool one gets overall $n + m$ results for $(\rho * n) + (m + n)$ requests, i.e. an efficiency of $\frac{n+m}{\rho * n+m+n}$. To illustrate the potential difference in efficiencies, we give the following example. In the approach without challenges, for $\rho = 3$ one gets an efficiency of $\frac{1}{3} \approx 0.333$. In the "hybrid" approach with redundancy and challenges, using $\rho = 3$, one can get an efficiency of $\frac{6}{15} = 0.4$ (for $m = n = 3$) or $\frac{8}{14} \approx 0.571$ (for $m = 6, n = 2$). The more an agent trusts a provider, the less challenges he can use and so the higher the efficiency will be.

[1] This approach would mainly improve the use of redundant requests in terms of efficiency, but still would be partially sensitive to collusion.

5 Related Work

Several trust and reputation models base their computations on a beta distribution [13,12,21,22]. However, they all assume that the assessing agent is able to verify what we call the "real requests". Our mechanism addresses the cases in which this verification is undesirable or not possible. These cases are numerous which was illustrated in Section 2.

Fullam et al. [23] propose a method for information acquisition from possibly malicious or incompetent sources. Also, they show how to manage the trade-off between costs for information acquisition, quality of the acquired information and the coverage of an agent's goals. In their model, the reliability of an information provider is assessed by checking whether some acquired information fits the agent's beliefs. In contrast to our mechanism, their approach does not allow to handle acquired information that is not related to an agent's beliefs (e.g. mathematical calculations, music data, etc.).

Liau[24] models the relationship among *belief*, *trust* and *information acquisition* by use of modal logics. They formally study the role that trust can play when uncertain information is assimilated in an agent's beliefs.

In network security, a family of authentication protocols uses the principle of *challenge-response* (e.g. [25,26]). Here, an entity proves its identity by answering to a challenge posed by the opponent – this challenge can only be answered, if the entity is in possession of a certain secret, and this secret is only given to the entity with the identity in claim. However, our mechanism is not related to this class of protocols: We do not use secrets and moreover, in our case it is not about authentication.

6 Conclusion and Future Work

In this paper, a trust-based mechanism was presented which uses challenges to estimate the correctness of acquired information that cannot be verified. We showed how to choose an optimal number of challenges for a given number of real requests and found that it is advantageous if the number of challenges is a multiple of the number of real requests, or divides it. A way to reduce this optimal number of challenges was proposed that makes use of *trust*. For these purposes, a formal trust-model was introduced that computes trust-values and uncertainty based on the beta distribution. It was proven that our uncertainty measure preserves the properties demanded by Wang and Singh [12] but is easier to compute.

Currently, we are working on a bootstrapping procedure for the mechanism. Besides, we develop a procedure for deciding whether some acquired information is accurate enough or should be requested again from other agents. Finally, the mechanism is planned to be implemented and tested.

Acknowledgments. We would like to thank Ulrich Sorger for many valuable discussions and Uwe Roth and Daniel Fischer for their constructive feedbacks.

References

1. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24(1), 33–60 (2005)
2. Ramchurn, S.D., Huynh, T.D., Jennings, N.R.: Trust in multi-agent systems. Knowl. Eng. Rev. 19(1), 1–25 (2004)
3. Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M., Werthimer, D.: SETI@home: An experiment in public-resource computing. Commun. ACM 45(11), 56–61 (2002)
4. Rehák, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 298–312. Springer, Heidelberg (2007)
5. Berman, F., Fox, G., Hey, A.J.G.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons, Inc., New York (2003)
6. Weiss, A.: Computing in the clouds. netWorker 11(4), 16–25 (2007)
7. Toh, C.K.: Ad Hoc Wireless Networks: Protocols and Systems. Prentice Hall PTR, Upper Saddle River (2001)
8. Christin, N., Weigend, A.S., Chuang, J.: Content availability, pollution and poisoning in file sharing peer-to-peer networks. In: EC 2005: Proc. of the 6th ACM Conf. on Electronic commerce, pp. 68–77. ACM Press, New York (2005)
9. Abdul-Rahman, A., Hailes, S.: Supporting trust in virtual communities. In: HICSS 2000: Proc. of the 33rd Hawaii Int. Conf. on System Sciences. IEEE Computer Society Press, Los Alamitos (2000)
10. Rohatgi, V.K.: Statistical Inference. Dover Publications, Incorporated, Mineola (2003)
11. Kinateder, M., Baschny, E., Rothermel, K.: Towards a generic trust model - comparison of various trust update algorithms. In: Herrmann, P., Issarny, V., Shiu, S.C.K. (eds.) Trust 2005. LNCS, vol. 3477, pp. 177–192. Springer, Heidelberg (2005)
12. Wang, Y., Singh, M.P.: Formal trust model for multiagent systems. In: IJCAI 2007: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, pp. 1551–1556 (2007)
13. Teacy, W.T.L., Patel, J., Jennings, N.R., Luck, M.: Travos: Trust and reputation in the context of inaccurate information sources. Auton. Agents Multi-Agent Syst. 12(2), 183–198 (2006)
14. Jaynes, E.T.: Probability Theory: The Logic of Science. Cambridge University Press, Cambridge (2003)
15. Montgomery, D.C.: Introduction to Statistical Quality Control, 5th edn. John Wiley, Chichester (2004)
16. Dumais, S.T., Furnas, G.W., Landauer, T.K., Deerwester, S., Harshman, R.: Using latent semantic analysis to improve access to textual information. In: CHI 1988: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems, pp. 281–285. ACM, New York (1988)
17. Hofmann, T.: Probabilistic latent semantic indexing. In: SIGIR 1999: Proc. of the 22nd annual Int. ACM SIGIR Conf. on Research and Development in Information Retrieval, pp. 50–57. ACM, New York (1999)
18. Karypis, G., Han, E.: Concept indexing: A fast dimensionality reduction algorithm with applications to document retrieval and categorization. Technical Report TR-00-0016, University of Minnesota (2000)
19. Dearden, R., Friedman, N., Andre, D.: Model based bayesian exploration. In: UAI 1999: Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence, pp. 150–159 (1999)

20. Chalkiadakis, G., Boutilier, C.: Coordination in multiagent reinforcement learning: a bayesian approach. In: AAMAS 2003: Proc. of the 2nd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp. 709–716. ACM Press, New York (2003)
21. Buchegger, S., Boudec, J.Y.L.: A robust reputation system for mobile ad hoc networks. Technical Report IC/2003/50, EPFL-IC-LCA, CH-1015 Lausanne (July 2003)
22. Jøsang, A., Ismail, R.: The beta reputation system. In: Proc. of the 15th Bled Conf. on Electronic Commerce, pp. 324–337 (2002)
23. Fullam, K.K., Park, J., Barber, K.S.: Trust-driven information acquisition for secure and quality decision-making. In: KIMAS 2005: Proc. of Int. Conf. on Integration of Knowledge Intensive Multi-Agent Systems, pp. 303–310 (2005)
24. Liau, C.J.: Belief, information acquisition, and trust in multi-agent systems: a modal logic formulation. Artif. Intell. 149(1), 31–60 (2003)
25. Otway, D., Rees, O.: Efficient and timely mutual authentication. SIGOPS Oper. Syst. Rev. 21(1), 8–10 (1987)
26. Steiner, J.G., Neuman, C., Schiller, J.I.: Kerberos: An authentication service for open network systems. In: Proc. of the Winter 1988 Usenix Conference, pp. 191–2024 (1988)

Modeling Dynamics of Relative Trust of Competitive Information Agents

Mark Hoogendoorn, S. Waqar Jaffry, and Jan Treur

Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
{mhoogen,swjaffry,treur}@few.vu.nl

Abstract. In order for personal assistant agents in an ambient intelligence context to provide good recommendations, or pro-actively support humans in task allocation, a good model of what the human prefers is essential. One aspect that can be considered to tailor this support to the preferences of humans is trust. This measurement of trust should incorporate the notion of relativeness since a personal assistant agent typically has a choice of advising substitutable options. In this paper such a model for relative trust is presented, whereby a number of parameters can be set that represent characteristics of a human.

1 Introduction

Nowadays, more and more ambient systems are being deployed to support humans in an effective way [1], [2] and [3]. An example of such an ambient system is a personal agent that monitors the behaviour of a human executing certain complex tasks, and gives dedicated support for this. Such support could include advising the use of a particular information source, system or agent to enable proper execution of the task, or even involving such a system or agent pro-actively. In order for these personal agents to be accepted and useful, the personal agent should be well aware of the habits and preferences of the human it is supporting. If a human for example dislikes using a particular system or agent, and there are several alternatives available that are more preferred, the personal agent would not be supporting effectively if it would advise, or even pro-actively initiate, the disliked option.

An aspect that plays a crucial role in giving such tailored advice is to represent the trust levels the human has for certain options. Knowing these trust values allows the personal assistant to reason about these levels, and give the best possible support that is in accordance with the habits and preferences of the human. Since there would be no problem in case there is only one way of supporting the human, the problem of selecting the right support method only occurs in case of substitutable options. Therefore, a notion of relative trust in these options seems more realistic than having a separate independent trust value for each of these options. For instance, if three systems or agents can contribute X, and two of them perform bad, whereas the third performs pretty bad as well, but somewhat better in than the others, your trust in that third option may still be a bit high since in the context of the other options it is the best alternative. The existing trust models do however not explicitly handle such relative trust notions [4] and [5].

This paper introduces an approach to model relative trust. In this model, a variety of different parameters can be set to fully tailor this trust model towards the human being supported. These aspects include initial trust and distrust, the weighing of positive and negative experiences, and the weight of past experiences. The model is represented by means of differential equations to also enable a formal analysis of the proposed model. Experiments have been conducted with a variety of settings to show what the influence of the various parameters is upon the trust levels.

This paper is organised as follows. First, in Section 2 the model is explained. Next, Section 3 presents a number of simulation results. In Section 4 the model is used to compare different cultures with each other. Section 5 presents a formal analysis of the model. Finally, Section 6 is a discussion.

2 Modelling Dynamics of Trust of Competitive Trustees

This section proposes a model that caters the dynamics of a human's trust on competitive trustees. In this model trust of the human on a trustee depends on the relative experiences with the trustee in comparison to the experiences from all of the competitive trustees. The model defines the total trust of the human as the difference between positive trust and negative trust (distrust) on the trustee. It includes personal human characteristics like trust decay, flexibility, and degree of autonomy (context-independence) of the trust. Figure 1 shows the dynamic relationships in the proposed model.

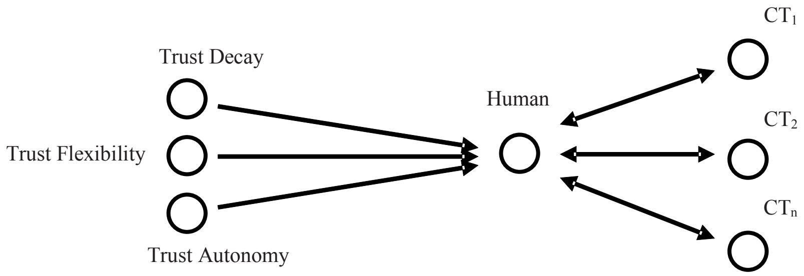


Fig. 1. Trust-based interaction with n competitive trustees (information agents IA)

In this model it is assumed that the human is bound to request one of the available competitive trustees at each time step. The probability of the human's decision to request one of the trustees $\{CT_1, CT_2, \ldots CT_n\}$ at time t is based on the trust value $\{T_1, T_2, \ldots T_n\}$ for each CT_i respectively at time t. In the response of the human's request CT_i gives experience value ($E_i(t)$) from the set {-1, 1} which means a negative and positive experience respectively. This experience is used to update the trust value for the next time point. Besides {-1, 1} the experience value can also be 0, indicating that CT_i gives no experience to the human at time point t.

2.1 Parameters Characterising Individual Differences between Humans

To tune the model to specific personal human characteristics a number of parameters are used.

Flexibility β. The personality attribute called trust flexibility (β) is a number between [0, 1] that represents in how far the trust level at time point t will be adapted when human has a (positive or negative) experience with a trustee. If this factor is high then the human will give more weight to the experience at $t+\Delta\ t$ than the already available trust at t to determine the new trust level for $t+\Delta\ t$ and vice versa.

Trust Decay γ. The human personality attribute called trust decay (γ) is a number between [0, 1] that represents the rate of trust decay of the human on the trustee when there is no experience. If this factor is high then the human will forget soon about past experiences with the trustee and vice versa.

Autonomy η. The human personality attribute called autonomy (η) is a number between [0, 1] that indicates in how far trust is determined independent of trust in other options. If the number is high, trust is (almost) independent of other options.

Initial Trust. The human personality attribute called initial trust indicates the level of trust assigned initially to a trustee.

2.2 Dynamical Models for Relative Trust and Distrust

The model is composed from two models: one for the positive trust, accumulating positive experiences, and one for negative trust, accumulating negative experiences. The approach of taking positive and negative trust separately at the same time to measure total trust is similar to the approaches taken in literature for degree of belief and disbelief [6] and [7]. Both negative and positive trusts are a number between [0, 1]. While human total trust at CT_i on any time point t is the difference of positive and negative trust at CT_i at time t.

Here first the positive trust is addressed. The human's relative positive trust of CT_i at time point t is based on a combination of two parts: the *autonomous* part, and the *context-dependent* part. For the latter part an important indicator is the human's relative positive trust of CT_i at time point t (denoted by $\tau_i^+(t)$): the ratio of the human's trust of CT_i to the average human's trust on all options at time point t. Similarly an indicator for the human's relative negative trust of CT_i at time point t (denoted by $\tau_i^-(t)$) is the ratio between human's negative trust of the option CT_i and the average human's negative trust on all options at time point t. These are calculated as follows:

$$\tau_i^+(t) = \frac{T_i^+(t)}{\sum_{j=1}^{n} T_j^+(t) \Big/ n} \quad \text{and} \quad \tau_i^-(t) = \frac{T_i^-(t)}{\sum_{j=1}^{n} T_j^-(t) \Big/ n}$$

Here the denominators $\sum_{j=1}^{n} T_j^+(t) \big/ n$ and $\sum_{j=1}^{n} T_j^-(t) \big/ n$ express the average positive and negative trust over all options at time point t respectively. The context-dependent part was designed in such a way that when the positive trust is above the average, then upon each positive experience it gets an extra increase, and when it is below average it gets a decrease. This models a form of competition between the different information agents. The principle used is a variant of a 'winner takes it all' principle, which for example is sometimes modelled by mutually inhibiting neurons representing the different options. This principle has been modelled by basing the change of trust upon a positive experience on $\tau_i^+(t) - 1$, which is positive when the positive trust is above average and negative when it is below average. To normalise, this is multiplied by a factor $T_i^+(t)*(1 - T_i^+(t))$. For the autonomous part the change upon a positive experience is modelled by $1 - T_i^+(t)$. As η indicates in how far the human is autonomous or context-dependent in trust attribution, a weighted sum is taken with weights η and $1\text{-}\eta$ respectively. Therefore, using the parameters defined in above $T_i^+(t+\Delta t)$ is calculated using the following equations. Note that here the competition mechanism is incorporated in a dynamical systems approach where the values of $\tau_i^+(t)$ have impact on the change of positive trust over time. Followings are the equations when $E_i(t)$ is 1, 0 and -1 respectively.

$$T_i^+(t+\Delta t) = T_i^+(t) + \beta * \left(\eta * \left(1 - T_i^+(t)\right) + (1-\eta) * \left(\tau_i^+(t) - 1\right) * T_i^+(t) * \left(1 - T_i^+(t)\right)\right) * \Delta t$$
$$T_i^+(t+\Delta t) = T_i^+(t) - \gamma * T_i^+(t) * \Delta t$$
$$T_i^+(t+\Delta t) = T_i^+(t)$$

Notice that here in the case of negative experience positive trust is kept constant to avoid doubling the effect over all trust calculation as negative experience is accommodated fully in the negative trust calculation. In one formula this is expressed by:

$$T_i^+(t+\Delta t) = T_i^+(t) + \left(\begin{array}{l} \beta * \left(\eta * \left(1 - T_i^+(t)\right) + (1-\eta) * \left(\tau_i^+(t) - 1\right) * T_i^+(t) * \left(1 - T_i^+(t)\right)\right) * E_i(t) \\ * \left(E_i(t) + 1\right)/2 - \gamma * T_i^+(t) * \left(1 + E_i(t)\right) * \left(1 - E_i(t)\right) \end{array}\right) * \Delta t$$

In differential equation form this can be reformulated as:

$$\frac{dT_i^+(t)}{dt} = \beta * \left(\eta * \left(1 - T_i^+(t)\right) + (1-\eta) * \left(\tau_i^+(t) - 1\right) * T_i^+(t) * \left(1 - T_i^+(t)\right)\right) * E_i(t) * \left(E_i(t) + 1\right)/2$$
$$- \gamma * T_i^+(t) * \left(1 + E_i(t)\right) * \left(1 - E_i(t)\right)$$

Notice that this is a system of n coupled differential equations; the coupling is realised by $\tau_i^+(t)$ which includes the sum of the different trust values for all j. Similarly, for negative trust followings are the equations when $E_i(t)$ is -1, 0 and 1 respectively.

$$T_i^-(t+\Delta t) = T_i^-(t) + \beta * \left(\eta * \left(1 - T_i^-(t)\right) + (1-\eta) * \left(\tau_i^-(t) - 1\right) * T_i^-(t) * \left(1 - T_i^-(t)\right)\right) * \Delta t$$
$$T_i^-(t+\Delta t) = T_i^-(t) - \gamma * T_i^-(t) * \Delta t$$
$$T_i^-(t+\Delta t) = T_i^-(t)$$

In one formula this is expressed as:

$$T_i^-(t+\Delta t) = T_i^-(t) + \left(\begin{array}{l} \beta * \left(\eta * \left(1 - T_i^-(t)\right) + (1-\eta) * \left(\tau_i^-(t) - 1\right) * T_i^-(t) * \left(1 - T_i^-(t)\right)\right) * E_i(t) \\ * \left(E_i(t) - 1\right)/2 - \gamma * T_i^-(t) * \left(1 + E_i(t)\right) * \left(1 - E_i(t)\right) \end{array}\right) * \Delta t$$

In differential equation form this can be reformulated as:

$$\frac{dT_i^-(t)}{dt} = \beta * \left(\eta * \left(1 - T_i^-(t)\right) + (1-\eta) * \left(\tau_i^-(t) - 1\right) * T_i^-(t) * \left(1 - T_i^-(t)\right)\right) * E_i(t) * \left(E_i(t) - 1\right)/2$$
$$- \gamma * T_i^-(t) * \left(1 + E_i(t)\right) * \left(1 - E_i(t)\right)$$

Notice that this again is a system of n coupled differential equations but not coupled to the system for the positive case described above.

2.3 Combining Positive and Negative Trust in Overall Relative Trust

The human's total trust $T_i(t)$ of CT_i at time point t is a number between [-1, 1] where -1 and 1 represent minimum and maximum values of the trust respectively. It is the difference of the human's positive and negative trust of CT_i at time point t:

$$T_i(t) = T_i^+(t) - T_i^-(t)$$

In particular, also the human's initial total trust of CT_i at time point 0 is $T_i(0)$ which is the difference of human's initial trust $T_i^+(0)$and distrust $T_i^-(0)$ in CT_i at time point 0.

2.4 Decision Model for Selection of a Trustee

As the human's total trust is a number in the interval [-1, 1], to calculate the *request probability* to request CT_i at time point t ($RP_i(t)$) the human's total trust $T_i(t)$ is first projected at the interval [0, 2] and then normalized as follows;

$$RP_i(t) = \frac{T_i(t) + 1}{\sum_{j=1}^{n} \left(T_j(t) + 1\right)}.$$

3 Simulation Results

This section describes a case study to analyze the behavior of the model described in Section 2. This case study analyzes the dynamics of a human's total trust on the three competitive Information Agents (IA's). Several simulations were conducted in this case study. Few of the simulation results are presented in this and the next section. Other variations could be found in appendix A[1]. In this case study it is assumed that the human is bound to request one of the available competitive information agents at each time step. The probability of the human's decision to request one of the information agents $\{IA_1, IA_2, IA_3\}$ at time t is based on the human's total trust with each information agent respectively at time t $\{T_1(t), T_2(t), T_3(t)\}$ (i.e. the equation shown in Section 2.4). In response of the human's request for information the agent gives an experience value $E_i(t)$.

3.1 Relativeness

The first experiment described was conducted to observe the relativeness attribute of the model (see Figure 2). In the Figure, the x-axis represents time, whereas the y-axis represents the trust value for the various information providers. The configurations taken into the account are as shown in Table 1.

It is evident from above graphs that the information agent who gives more positive experience gets more relative trust than the others, which can be considered a basic property of trust dynamics (trust monotonicity) [5] and [8].

[1] http://www.cs.vu.nl/~mhoogen/trust/appendix-CIA-2008.pdf

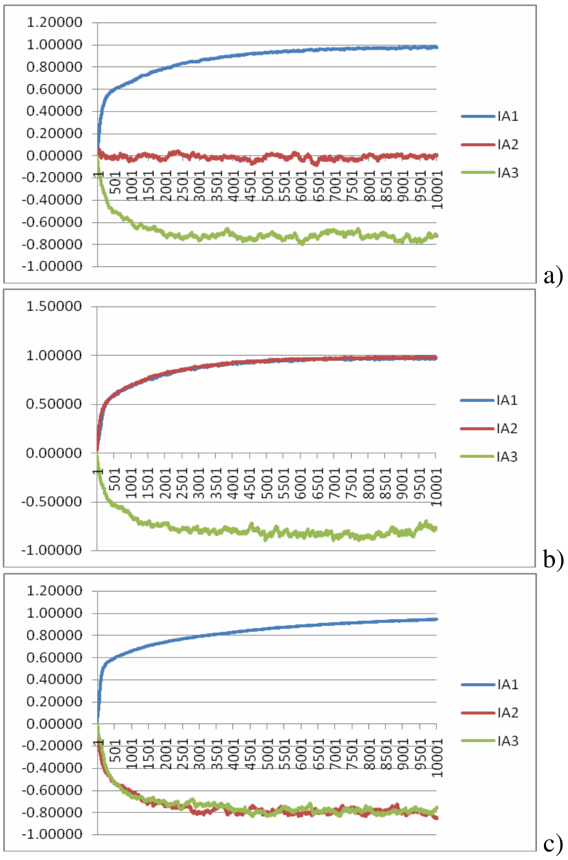


Fig. 2. Model Dependence on Amount of Positive Response from IAs: a) Information Agents IA_1, IA_2, IA_3 give experience positive, random (equal probability to give a positive or negative experience), negative respectively on each request by the Human respectively. b) Information Agents IA_1, IA_2, IA_3 give experience positive, positive, negative on each request by the Human respectively. c) Information Agents IA_1, IA_2, IA_3 give experience positive, negative, negative on each request by the Human respectively.

Table 1. Parameter values to analyze the dynamics of relative trust with the change in IAs responses

Attribute	Symbol	Value
Trust Decay	γ	0.01
Autonomy	η	0.25
Flexibility	β	0.75
Time Step	Δt	0.10
Initial Trust and Distrust of $\{IA_1, IA_2, IA_3\}$	$T_1^+(0)$, $T_2^+(0)$, $T_3^+(0)$, $T_1^-(0)$, $T_2^-(0)$, $T_3^-(0)$	0.50, 0.50, 0.50, 0.50, 0.50, 0.50

3.2 Trust Decay

This second experiment, shown in Figure 3, was configured to observe the change in the total trust in relation to change in the trust decay attribute γ of the human. The configurations taken into the account are as shown in Table 2.

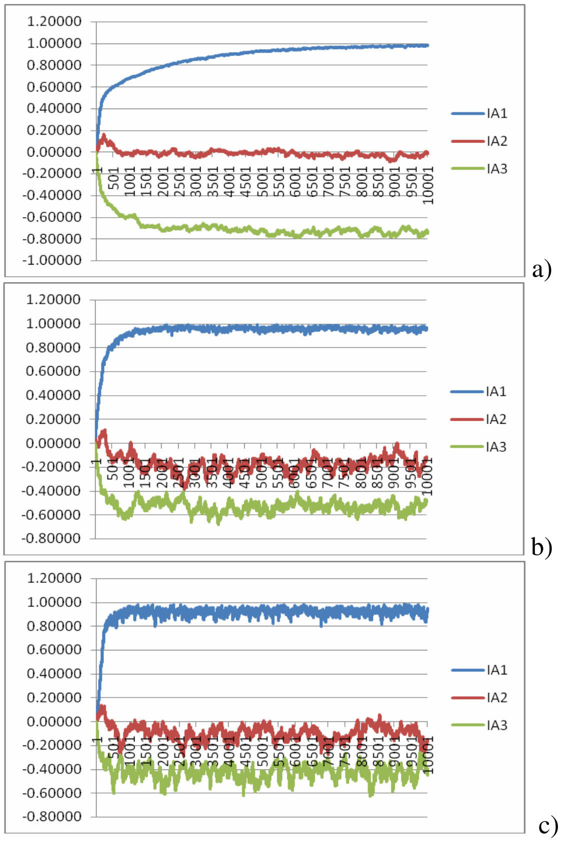


Fig. 3. Model Dependence on Trust Decay: a) $\gamma = 0.01$. b) $\gamma = 0.05$. c) $\gamma = 0.10$.

Table 2. Parameter values to analyze the dynamics of relative trust with the change in trust decay (γ)

Attribute	Symbol	Value
Experience {IA_1, IA_2, IA_3}	E_1, E_2, E_3	1, random, -1
Autonomy	η	0.25
Flexibility	β	0.75
Time Step	Δt	0.10
Initial Trust and Distrust of {IA_1,IA_2,IA_3}	$T_1^+(0)$, $T_2^+(0)$, $T_3^+(0)$, $T_1^-(0)$, $T_2^-(0)$, $T_3^-(0)$	0.50,0.50,0.50, 0.50,0.50,0.50

In these cases also the information agent who gives more positive experience gets more relative trust than the others. Furthermore, if the trust decay is higher, then the trust value drops rapidly on no experience (see Figure 3c; more unsmooth fringes of the curve).

3.3 Flexibility of Trust

This experiment is configured to observe the change in the total trust with the change in the human's flexibility of the trust (see Figure 4). Configurations taken into the account are shown in Table 3.

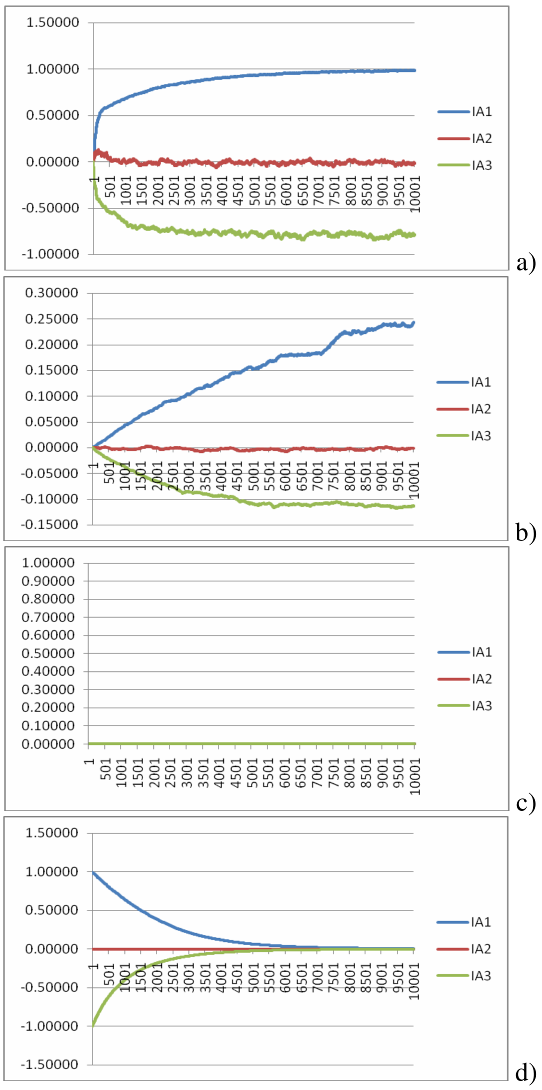


Fig. 4. Model Dependence on Trust Flexibility: a) $\beta = 1$, b) $\beta = 0.01$, c) $\beta = 0.00$, d) $\beta = 0.00$ and $T_1(0)=1$, $T_2(0)=0$, $T_3(0)=-1$

Table 3. Parameter values to analyze the dynamics of relative trust with the change in flexibility (β)

Attribute	Symbol	Value
Experience {IA_1, IA_2, IA_3}	E_1, E_2, E_3	1, random, -1
Trust Decay	γ	0.01
Autonomy	η	0.25
Time Step	Δt	0.10
Initial Trust and Distrust of {IA_1,IA_2,IA_3}	$T_1^+(0)$, $T_2^+(0)$, $T_3^+(0)$, $T_1^-(0)$, $T_2^-(0)$, $T_3^-(0)$	0.50, 0.50, 0.50, 0.50, 0.50, 0.50

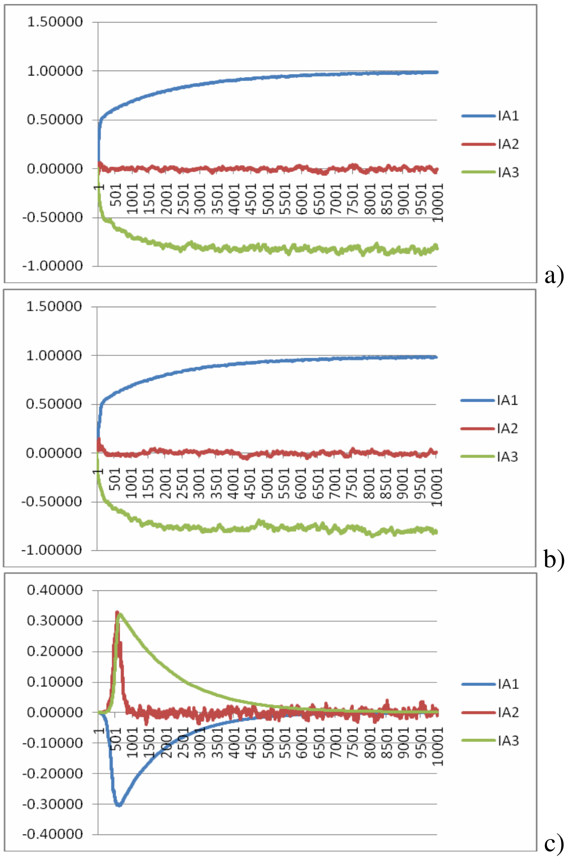


Fig. 5. Model Dependence on Trust Autonomy: a) η=1.0, b) η=0.50, c) η=0.00

In these cases again the information agent who gives more positive experience gets more human's relative trust then the others. Furthermore as the values of the β decrease the rate of change of the trust also decrease. In Figure 4c, β=0 which means that trust does not change on experiences at all, so the initial values retain for experiences from the information agents hence trust value remains stable. Finally in the Figure 4d as initial values of the total trust are taken $T_1(0)=1$, $T_2(0)=0$ and $T_3(0)=-1$ instead of $T_1(0)=0$, $T_2(0)=0$ and $T_3(0)=0$, so the total trust decays due to the trust decay factor and becomes stable after a specific time span.

3.4 Autonomy of Trust

This experiment (see Figure 5) is configured to observe the change in the human trust with the change in the human's autonomy for the total trust calculation. Configurations taken into the account are shown in Table 4.

In these cases also the information agent who gives more positive experience gets more relative trust then the others. Further more as the values of the η decrease the human weights the relative part of the trust more than the autonomous trust. In Figure 5c, $\eta=0$ which means that the human does not take into account the autonomous trust. This gives unstable patterns that are extremely sensitive to the initial conditions of the system. The example graph shown is just one of these patterns.

Table 4. Parameter values to analyze the dynamics of relative trust with the change in autonomy (η)

Attribute	Symbol	Value
Experience {IA_1, IA_2, IA_3}	E_1, E_2, E_3	1, random, -1
Trust Decay	γ	0.01
Flexibility	β	0.75
Time Step	Δt	0.10
Initial Trust and Distrust of {IA_1,IA_2,IA_3}	$T_1^+(0)$, $T_2^+(0)$, $T_3^+(0)$, $T_1^-(0)$, $T_2^-(0)$, $T_3^-(0)$	0.50, 0.50, 0.50, 0.50, 0.50, 0.50

3.5 Initial Trust and Distrust

This experiment is configured to observe the change in the total trust with the change in the human's initial trust and distrust ($T^+_i(0)$, $T^-_i(0)$) on information agents (see Figure 6). Configurations taken into the account are shown in Table 5.

Table 5. Parameter values to analyze the dynamics of relative trust with the change in initial trust

Attribute	Symbol	Value
Experience {IA_1, IA_2, IA_3}	E_1, E_2, E_3	1, random, -1
Trust Decay	γ	0.01
Autonomy	η	0.25
Flexibility	β	0.75
Time Step	Δt	0.10

It is observed from the above graphs that the final outcome of the trust is not very sensitive for the initial values.

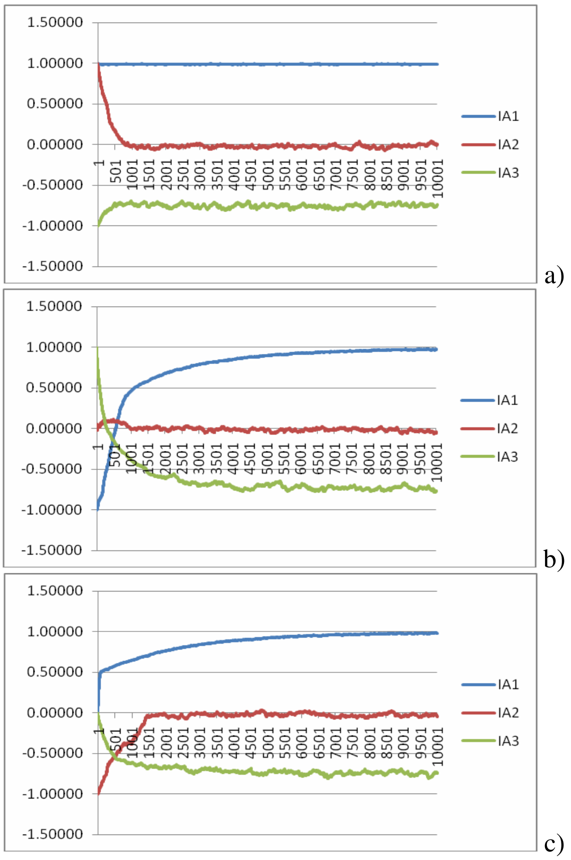


Fig. 6. Model Dependence on Initial Trust {$T_1(0)$, $T_2(0)$, $T_3(0)$}: a) 1, 1, -1. b) -1, 0, 1. c) 0, -1, 0

4 Dynamics of Relative Trust in Different Cultures

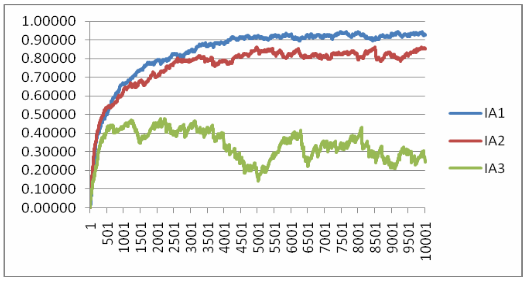
The degree of reliability of available information sources may strongly differ in different types of societies or cultures. In some types of societies it may be exceptional when an information source provides 10% or more false information, whereas in other types of societies it is more or less normal that around 50% of the outcomes of information sources is false. If the positive experiences percentage given by the information

Table 6. Classification of Human Cultures with respect to the Positive Experiences given by the IAs

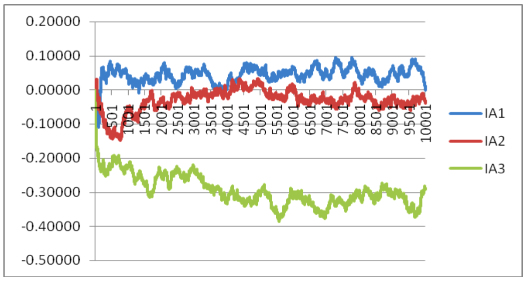
Culture Name	Percentage of the positive experiences by the information agents {IA_1, IA_2, IA_3}
A	100, 99, 95
B	50, 40, 30
C	10, 0, 0
D	0,0,0

Table 7. Parameter values to analyze the Relative Trust Dynamics in different Cultures

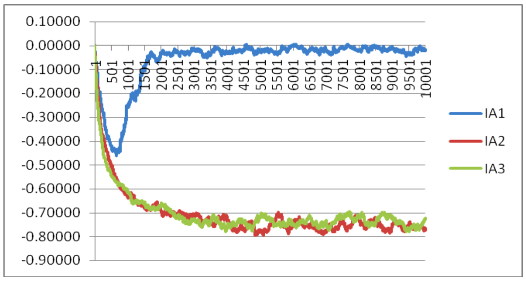
Attribute	Symbol	Value
Trust Decay	γ	0.01
Autonomy	η	0.25
Flexibility	β	0.75
Time Step	Δt	0.10
Initial Trust and Distrust of $\{IA_1, IA_2, IA_3\}$	$T_1^+(0)$, $T_2^+(0)$, $T_3^+(0)$, $T_1^-(0)$, $T_2^-(0)$, $T_3^-(0)$	0.50,0.50,0.50, 0.50,0.50,0.50



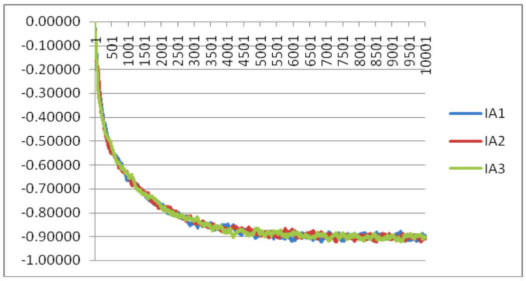
a)



b)



c)



d)

Fig. 7. Dynamics of Relative Trust in Different Cultures. a) Culture A, b) Culture B, c) Culture C, d) Culture D.

agents varies significantly, then the total relative trust of the human on the these information agents may differ as well. This case study was designed to study dynamics of the human's trust on information agents in different cultures with respect to the percentages of the positive experiences they provide to the human. A main question is whether in a culture where most information sources are not very reliable, the trust in a given information source is higher than in a culture where the competitive information sources are more reliable. Cultures are named with respect to percentage of the positive experiences provided by the information agents to the human as shown in Table 6 and other experimental configurations in Table 7.

Simulation results for the dynamics of the relative trust for the cultures mentioned in Table 6 are shown in Figure 7.

From Figure 7 it can be concluded that in every culture whatever relative percentage of the positive experiences may be (except when all information agent give negative experiences all of the time (see Figure 7d), the information agent that gives more positive experiences to the human gains more trust. Furthermore, the information agent that gives more positive experiences at least secure neutral trust (*T(t)=0*) in the long run, even the percentage of positive experiences is very low (see Figure 7c).

5 Formal Analysis of the Model

In this section a mathematical analysis is made of the change in trust upon positive (resp. negative) experiences. In Section 2 the differential equation form of the model for positive trust was formulated as:

$$\frac{dT_i^+(t)}{dt} = \beta * \left[\eta * \left(1 - T_i^+(t)\right) + (1-\eta) * \left(\tau_i^+(t) - 1\right) * T_i^+(t) * \left(1 - T_i^+(t)\right)\right] * E_i(t) * \left(E_i(t) + 1\right)/2$$
$$- \gamma * T_i^+(t) * \left(1 + E_i(t)\right) * \left(1 - E_i(t)\right)$$

where $\tau_i^+(t)$ is

$$\tau_i^+(t) = \frac{T_i^+(t)}{\sum_{j=1}^{n} T_j^+(t) \big/ n}$$

One question that can be addressed is when for a given time point *t* an equilibrium occurs, i.e. under which conditions trust does not change at time point *t*. Another question is under which circumstances trust will increase at *t*, and under which it will decrease. As the experience function $E_i(t)$ is given by an external scenario, these questions have to be answered for a given value of this function. So, three cases are considered:

Case 1: $E_i(t) = 1$

In this case the differential equation can be simplified to

$$\frac{dT_i^+(t)}{dt} = \beta * \left(\eta * \left(1 - T_i^+(t)\right) - (1-\eta) * \left(1 - \tau_i^+(t)\right) * T_i^+(t) * \left(1 - T_i^+(t)\right)\right)$$

$$\frac{dT_i^+(t)}{dt} = \beta * \left(\eta - (1-\eta) * \left(1 - \frac{T_i^+(t)}{\sum_{j=1}^{n} T_j^+(t) \big/ n}\right) * T_i^+(t)\right) * \left(1 - T_i^+(t)\right)$$

It follows that $\frac{dT_i^+(t)}{dt} \geq 0$ if and only if

$$\eta - (1-\eta) * \left(1 - \frac{T_i^+(t)}{\sum_{j=1}^n T_j^+(t) / n}\right) * T_i^+(t) \geq 0$$

or

$$T_i^+(t) = 1$$

For $T_i^+(t) < 1$ this is equivalent to (with: $S(t) = \sum_{j=1}^n T_j^+(t)$)

$$(1-\eta) * \left(1 - \frac{T_i^+(t)}{S(t)/n}\right) * T_i^+(t) \leq \eta$$
$$(1-\eta) * (S(t) - n * T_i^+(t)) * T_i^+(t) \leq \eta * S(t)$$
$$(S(t) * T_i^+(t) - n * T_i^+(t)^2) \leq \eta * S(t)/(1-\eta)$$
$$n * T_i^+(t)^2 - S(t) * T_i^+(t) + \eta * S(t)/(1-\eta) \geq 0$$

This quadratic expression in $T_i^+(t)$ has no zeros when the discriminant $S(t)^2 - \frac{4n * S(t) * \eta}{(1-\eta)}$ is negative:

$$S(t)^2 - \frac{4n * S(t) * \eta}{(1-\eta)} < 0 \Leftrightarrow S(t)\left(S(t) - \frac{4n * \eta}{(1-\eta)}\right) < 0 \Leftrightarrow 0 < S(t)/n < \frac{4\eta}{1-\eta}$$

When $\eta > 0.2$ then $1/\eta < 5$ and therefore $1/\eta - 1 < 4$, hence $(1-\eta)/\eta < 4$ which can be reformulated as $4\eta/(1-\eta) > 1$. As $S(t)/n \leq 1$, this shows that for $\eta > 0.2$ as long as $S(t)$ is positive, the discriminant is always negative, and therefore upon a positive experience there will always be an increase. When $S(t) = 0$, which means all trust values are 0, no change occurs. For the case the discriminant is ≥0, i.e., $S(t)/n \geq 4\eta/(1-\eta)$ then the quadratic equation $T_i^+(t)$ for has two zeros symmetric in $S(t)$:

$$T_i^+(t) = \left(S(t) +/- \sqrt{S(t)^2 - \frac{4n * S(t) * \eta}{1-\eta}}\right) \Big/ 2n$$

In this case increase upon a positive experience will take place for $T_i^+(t)$ less than the smaller zero or higher than the larger zero, and not between the zeros. An equilibrium occurs upon a positive experience when $T_i^+(t) = 1$ or when equality holds:

$$n * T_i^+(t)^2 - S(t) * T_i^+(t) - \eta * S(t)/(1-\eta) = 0$$

This only can happen when the discriminant is not negative, in which case equilibria occur for $T_i^+(t)$ equal to one of the zeros.

Case 2: $E_i(t) = 0$
In this case the differential equation can be simplified to

$$\frac{dT_i^+(t)}{dt} = -\gamma * T_i^+(t)$$

So, in this case positive trust is decreasing or has in equilibrium with positive trust 0.

Case 3: $E_i(t) = -1$
In this case the differential equation can be simplified to

$$\frac{dT_i^+(t)}{dt} = 0$$

So, for this case always an equilibrium occurs in t for positive trust.

For negative trust, the situation is a mirror image of the case for positive trust, and by combining the positive and negative trust, the patterns for overall trust can be analysed.

6 Discussion

This paper has introduced a model for relative trust to enable personal assistant agents to give the appropriate support to humans. Within the model several parameters have been introduced to tailor it towards a particular human. The influence of these parameters upon the trust has been extensively shown in this paper by means of simulations, even including different cultural settings. Finally, a mathematical analysis has been conducted to formally derive what the change of the trust functions is in case of positive and negative experiences.

A variety of trust models have been proposed in the literature [4] and [5]. These trust models attempt to determine the level of trust in certain agents based upon experiences. They do however not take into account the notion of relativeness of this trust. Models have been proposed for relative trust as well. In [9] a model is presented that allows an agent to combine multiple sources for deriving a trust value. This notion of relativeness differs from the notion used in this paper. [10] extends an existing trust model of [11] with the notion of relative trust. They take as a basis certain trust values determined by the model [11], and compare these values in order to make statements about different trust values for different agents. In determining the trust itself, they do not incorporate the experiences with other agents that can perform similar tasks, which is done in this paper. In [12] a trust model is utilized to allocate decision support tasks. In the model, relative trust is addressed as well but again not incorporated in the calculation of the trust value itself.

For future work, an interesting option is to see how well the parameters of the model can be derived by a personal assistant (based upon the requests outputted by the human).

References

1. Aarts, E., Harwig, R., Schuurmans, M.: Ambient Intelligence. In: Denning, P. (ed.) The Invisible Future, pp. 235–250. McGraw Hill, New York (2001)
2. Aarts, E., Collier, R., van Loenen, E., de Ruyter, B.: Ambient Intelligence. Proc. of the First European Symposium, EUSAI 2003. LNCS, vol. 2875, p. 432. Springer, Heidelberg (2003)
3. Riva, G., Vatalaro, F., Davide, F., Alcañiz, M. (eds.): Ambient Intelligence. IOS Press, Amsterdam (2005)
4. Falcone, R., Castelfranchi, C.: Trust dynamics: How trust is influenced by direct experiences and by trust itself. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), pp. 740–747 (2004)

5. Marx, M., Treur, J.: Trust Dynamics Formalised in Temporal Logic. In: Chen, L., Zhuo, Y. (eds.) Proc. of the Third International Conference on Cognitive Science, ICCS 2001, pp. 359–363. USTC Press, Beijing (2001)
6. Shortliffe, E.H., Buchanan, B.G.: A model of inexact reasoning in medicine. Mathematical Biosciences 23(3-4), 351–379 (1975)
7. Luger, G.F., Stubblefield, W.A.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving, 4th edn., pp. 320–321. Addison-Wesley, Reading (1998)
8. Jonker, C.M., Treur, J.: Formal Analysis of Models for the Dynamics of Trust based on Experiences. In: Garijo, F.J., Boman, M. (eds.) MAAMAW 1999. LNCS, vol. 1647, pp. 221–232. Springer, Heidelberg (1999)
9. Beth, T., Borcherding, M., Klein, B.: Valuation of trust in open networks. In: Gollmann, D. (ed.) ESORICS 1994. LNCS, vol. 875, pp. 3–18. Springer, Heidelberg (1994)
10. Kluwer, J., Waaler, A.: Relative Trustworthiness. In: Dimitrakos, T., Martinelli, F., Ryan, P.Y.A., Schneider, S. (eds.) FAST 2005. LNCS, vol. 3866, pp. 158–170. Springer, Heidelberg (2006)
11. Jones, A.: On the concept of trust. Decision Support Systems 33, 225–232 (2002)
12. van Maanen, P.-P., van Dongen, K.: Towards Task Allocation Decision Support by means of Cognitive Modeling of Trust. In: Castelfranchi, C., Barber, S., Sabater, J., Singh, M. (eds.) Proceedings of the Eighth International Workshop on Trust in Agent Societies (Trust 2005), pp. 168–177 (2005)

A Formal Approach to Aggregated Belief Formation

Annerieke Heuvelink[1,2], Michel C.A. Klein[1], and Jan Treur[1]

[1] Vrije Universiteit Amsterdam, Department of Artificial Intelligence,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
[2] TNO Defence, Security, and Safety, Department of Training and Instruction,
P.O. Box 23, 3769 ZG Soesterberg, The Netherlands
{a.heuvelink,michel.klein,treur}@few.vu.nl

Abstract. This paper introduces a formal method to aggregate over basic beliefs, in order to deduce aggregated or complex beliefs as often used in applications. Complex beliefs can represent several things, such as a belief about a period in which other beliefs held or the minimal or maximal certainty with which a belief held. As such they contain richer information than the basic beliefs they are aggregated from and can be used to optimize an agent's search through its memory and its reasoning processes. The developed method can also aggregate over aggregated beliefs, hence nested aggregations are possible. An implementation in Prolog demonstrates its operationality.

Keywords: Belief Aggregation, Term Algebra, Memory.

1 Introduction

Agents in applications commonly store beliefs about the state of the world in what is often called a world model or belief base. This belief base is usually a set of atomic beliefs that grows over time. There are several potential problematic issues related to such a belief base. First, after some time the size of the belief base can result in the practical problem that retrieval and inferences will become time expensive: the time needed will grow with the size of the belief base. Second, some inferences result in intermediate results that might be useful again at a later point in time. When the intermediate results are not stored, they have to be recalculated, while when they are stored, they will add to the size problem. Third, this way of storing beliefs seems not very similar to the human way of using memory. For example, humans often forget specific details, but still remember aggregated abstractions or consequences of specific facts. Taking this as a point of departure, it may be explored how aggregated beliefs can be formed and stored within an agent as new entities.

Fact is that aggregations can be formed from many different perspectives and at multiple levels. Which perspectives are chosen and which level of aggregation is needed, is application and task depended. Therefore a general approach that distinguishes all possible types of aggregations one by one, may become quite complex; for example, if *m* different aggregation types are possible, and *n* levels of aggregation, then the number of aggregation types is m^n, which already for relatively low numbers such as m = 10 and n = 5 leads to a high number (100.000) of aggregated beliefs. To avoid this explosion, in this paper an algebraic approach is adopted that distinguishes a general notion of aggregation operator that (1) is parameterised by the specific constraint that is used in an aggregation process, and (2) can be used in a recursive manner. Thus the combinatorics induced by different levels is replaced by term expressions that can be formed by nesting a number of (parameterised) variants of the aggregation operator.

The presented formalism allows for the specification of complex beliefs at a higher level of aggregation than the basic atomic beliefs. Such aggregated belief representations have the advantage that they are often closer to the level of aggregation that is used in specification of reasoning steps, and are therefore often more useful. For example, it is more convenient to specify reasoning steps based on an aggregated belief such as the "last most certain belief", than based on a long list of atomic beliefs at different time points and with different certainties. Moreover, some aggregations are used several times. In this case, the aggregation functions as a reasoning template that specifies how a new belief can be deduced from other beliefs. This template only has to be specified once and can be reused later on.

The remainder of the paper is structured as follows. First, section 2 introduces an application domain and the basic belief formalism that is used. In the subsequent section, the algebraic approach to belief aggregation is described, which is formalised as a term algebra in section 4. Section 5 and 6 demonstrate the operationality of the approach, by presenting a Prolog implementation and showing how the algebra allows the formation of several useful complex beliefs, which are defined as specific aggregations. Section 7 relates the work to other research, both in the area of knowledge compilation and temporal abstraction. Finally, in section 8 the research is summarised and future research plans exposed.

2 Belief Formalism

In [1] a software agent was developed that compiles a tactical picture of its environment, which entails the classification and identification of surface radar contacts. For modelling its behaviour the need was identified to explicitly represent the time at which a belief was held by the agent in its short term memory (STM). In other words: when it believed it. The main reason to represent this, was to enable the (biased) reasoning over (possible inconsistent) beliefs over time [1]. For example, when at time t it is believed that the position of a contact is [x1, y1], while at time t + n it is believed to be position [x2, y2], the average speed of the contact can be inferred. This is useful because the speed on a contact might contain information concerning its identity, e.g., large ships that are neutral usually do not sail faster than 20 knots. In the same way a contact's manoeuvring pattern can be inferred, which is relevant as it gives away much information concerning a contact's intent.

In order to logically represent other aspects, namely uncertainty of information, and the fact that information can come from various sources, every belief also received a source and a certainty label. As a result, the basic knowledge entity of the agent is represented by *belief(P, O, V, T, S, C)*, which denotes the belief that the indeterminable property P holds for object O with the value V at time T, based on source S and with certainty C. An example belief denoting that it is believed at time 8 with high certainty that the identity of the radar contact1 is friendly because of radio conservation is: *belief (identity, contact1, friendly, 8, radio, 0.9).*

The value, time, and certainty label of beliefs about a specific property and object are often used to reason about trends in those beliefs, which can lead to new beliefs. For example, a new belief can be formed about a contact being a merchant, and therefore neutral, due to beliefs about it sailing in a straight line. The certainty of the belief that the contact is a merchant is determined by the period, as by the certainty, with which it is believed that it does this. For the deduction of other beliefs it is often important to deduce what the last, or most (un)certain belief about a specific something was. For example, the highest certainty with which it was once believed that a contact fired is relevant for deducing whether it might be hostile.

The beliefs formed by the agent over time are stored in the agent's belief base, representing long term memory (LTM). When storing beliefs in LTM, it is important to denote when they were formed or retrieved in STM. For this a new reference to time is introduced, the two-place predicate *holds_at*. When the basic belief predicate of the object language is reified to a term *b*, the time at which the belief is held in STM can be expressed by *holds_at(b, t).* For every *belief(p, o, v, t, s, c)* that is found in the agent's belief base it holds that *holds_at(belief(p,o,v,t,s,c), t),* since the t of the belief denotes the time it was formed (was present in STM).

3 Belief Aggregation

Unfortunately, the storage of time-stamped beliefs led to the problematic issues mention in section 1 [2]. Therefore, this paper focuses on the development of a generic, formal approach to the formation of arbitrary aggregations over these basic beliefs, to form all kinds of so-called aggregated or complex beliefs. Complex beliefs abstract or cluster information of the lower level. They form a solution for keeping the amount of time required to search through the agent's belief base within limits. Furthermore, they can be used to model specific properties of human memory, like the forgetting of specific details.

3.1 Aggregation Examples

An example of a complex belief that an agent can form was mentioned in section 2, namely a belief about the period during which a certain belief held. That complex belief, about the duration of the straight manoeuvre of a contact, can be used directly to infer a new belief, e.g., that that contact might be a merchant.

While specifying the formal model underlying the reasoning and behaviour of the cognitive agent described in [1], it was found that often specific types of information are required to deduce new beliefs. To be precise, often the last, earliest, most certain or uncertain, increasing in certainty, or longest held belief was required. In addition, it was noticed that the deduction processes of several of these beliefs are very similar, e.g. the deduction of a last belief (belief with highest T) is very similar to the deduction of a most-certain belief (belief with highest C).

These observations spurred the development of an generic approach to belief aggregation in which a complex belief is defined as an aggregation that takes the form of a constraint (e.g., highest) that must hold for a certain variable (e.g., T) of a certain more of less specified belief (e.g., belief(identity, contact1, friendly, T, S, C) in which the P, O, and V are specified while the T, S, and C are left variable). The term algebra formalizing this approach to the formation of aggregated beliefs is introduced in the section 4, while the section after that discusses an implementation of the approach in Prolog.

3.2 Complex Belief of Type Integrated Sources

The *integrated_sources* belief was the most important complex belief the developed agent in [1] reasoned with instead of with its basic beliefs. This complex belief represents which value is currently believed by the agent to hold for a certain property and object, and with which certainty. To determine this, inconsistencies formed by beliefs from different sources, with different certainties, and held at different times, have to be resolved. Much research has been done on how to deal with such inconsistencies, see e.g. [3, 4].

In [1] a relative simple procedure was introduced to determine which value V was currently believed to held with certainty C by an agent for a given P and O. This procedure takes into account that a belief's validity over time is strongly influenced by its predicate type (property) P. Values of some predicates are much more persistent than others; consider the chance that a contact's position, speed, or intent changes over time. The following logical expression denotes the meaning of the complex belief called *integrated_sources*:

$$\begin{array}{l}
given\ (p, o) \\
\forall v1\ \forall t1\ \forall s1\ \forall c1\ \forall t\ \forall pd \qquad [\\
\qquad holds\_at(\ complex\_belief(\\
\qquad\quad integrated\_sources, for(p, o),\ has\_values(v1, c1 - pd * (t - t1))\ ), t) \\
\leftrightarrow \\
\qquad holds\_at(\ complex\_belief(\ last, for(p, o, s1), has\_values(v1, t1, c1)\ ), t) \wedge \\
\qquad persistence\_decay(p, pd) \wedge \\
\qquad \neg\exists s2\ \exists v2\ \exists t2\ \exists c2 \qquad [\\
\qquad\quad holds\_at(\ complex\_belief(\ last, for(p, o, s2), has\_values(v2, t2, c2)\ ), t) \wedge \\
\qquad\quad c2 - pd * (t - t2) > c1 - pd * (t - t1)\ ] \qquad]
\end{array} \qquad (1)$$

This expression specifies that the agent believes at time t that for a given P and O, *for(p, o)*, the value *v1* holds, which is the value of the belief about P and O whose certainty is the greatest after taking into account the time passed since it was formed and the persistence of the property; *c1 – pd * (t – t1)*. This might entail that the value of an older belief with a certain certainty is believed over the value of a newer belief that has a lower certainty. It might also be the other way around; it depends on the nature (persistence) of the property. In this expression another complex belief was used of the type *last*, which has as exact definition:

$$\begin{array}{l}
\forall p\ \forall o\ \forall v\ \forall t\ \forall s\ \forall c\ \forall n \qquad [\\
\qquad holds\_at(\ complex\_belief(\ last,\ for(p, o, s), has\_values(\ v, t, c)\ ), n) \\
\leftrightarrow \\
\qquad holds\_at(\ belief(p, o, v, t, s, c), t) \wedge t \leq n \wedge \\
\qquad \neg\exists t'\ \exists v'\ \exists c' \qquad [\\
\qquad\qquad holds\_at(\ belief(p, o, v', t', s, c'), t') \wedge t' \geq t \wedge t' \leq n\ ]\ ]
\end{array} \qquad (2)$$

Expression 2 specifies that the agent believes at time n that *t* is the last time at which a belief incorporating the given P, O, and S, *for(p, o, s)*, held. This is the case since *t* is the time label of a belief with that given P, O, and S, for which it holds that no other belief exists with the same P, O, and S, but a higher T (*t'*). This complex belief of type *last* is defined as an aggregation of all the beliefs with the given P, O, and S, and the constraint Highest for their time label T. Besides a specification for T, this aggregation also specifies the free variables V and C. This is a quite standard aggregation, considering the limited complexity of the constraint that it takes into account. The aggregation as which the complex belief of type *integrated_sources* is defined, is much less standard. The constraint that has to be taken into account in that aggregation is much more specific and not likely to be reusable, see section 5.

The current paper focuses on the development of an algebraic approach to have an efficient representation of aggregated beliefs. For demonstration purposes it elaborates on several possible types of these aggregated beliefs, which are defined as specific aggregations. Notice that the introduced aggregations simply serve as examples, and that many more are possible. The approach is set up in such a generic way that all kinds of constraints that lead to all kinds of complex beliefs can be expressed with it.

4 Algebraic Formalization

The algebra specification of the aggregation functions on beliefs is defined by a basic ontology, by means of which its objects and relations can be expressed. The primitive terms used in the algebra are defined by a many-sorted signature. The signature takes into account symbols for *sorts*, *constants*, *functions* and *relations*. Examples of *sorts* are: LABEL, CONSTRAINT, TIME, TYPE, AGGREGATIONBASE, AGGREGATEDBELIEF, ARGUMENTLIST, BASICBELIEFBASE, PROPERTY, or OBJECT. *Constants* are names of objects within sorts; examples are 'speed', '20', or 'fast'. *Functions* denote mappings from a (combination of) sort(s) to another sort; examples of function symbols are agg, holdsat, + and *. *Relations* symbols (relating different sorts) used are, for example = and <. Logical relationships involve conditional statements involving relations. Figure 1 depicts a large part or the algebra specification with the definitions used listed below. Arrows with no label are defined by 'e' which denotes (injective) embedding.

agg: LABEL x CONSTRAINT x AGGREGATEDBELIEF → AGGREGATIONNAME
e: AGGREGATIONBASE → AGGREGATEDBELIEF
e: BASICBELIEFBASE → AGGREGATEDBELIEF
e: COMPLEXBELIEFBASE → AGGREGATEDBELIEF
holdsat: AGGREGATIONNAME x TIME → AGGREGATIONBASE
definedas: COMPLEXBELIEFBASE x AGGREGATIONBASE
holdsat: COMPLEXBELIEF x TIME → COMPLEXBELIEFBASE
complexbelief: TYPE x ARGUMENTLIST x RANGELIST → COMPLEXBELIEF
e: PROPERTY → ARGUMENTLIST
e: OBJECT → ARGUMENTLIST
e: VALUE → ARGUMENTLIST
e: TIME → ARGUMENTLIST
e: SOURCE → ARGUMENTLIST

e: CERTAINTY → ARGUMENTLIST
e: RANGE → RANGELIST
holdsat: BASICBELIEF x TIME → BASICBELIEFBASE
belief: PROPERTY x OBJECT x VALUE x TIME x SOURCE x CERTAINTY → BASICBELIEF
abstraction1: LABELTYPE x VAR → LABEL
abstraction2: LABELTYPE x LABELTYPE x VAR x VAR → LABEL
constraint: NAME x VARIABLE x AGGREGATEDBELIEF → CONSTRAINT
forall, exists: VAR x FORMULA → FORMULA
definedas: CONSTRAINT x FORMULA
not: FORMULA → FORMULA
and, or, implies: FORMULA x FORMULA → FORMULA
e: ATOM → FORMULA
<, >, ≤, ≥ : TERM x TERM → ATOM

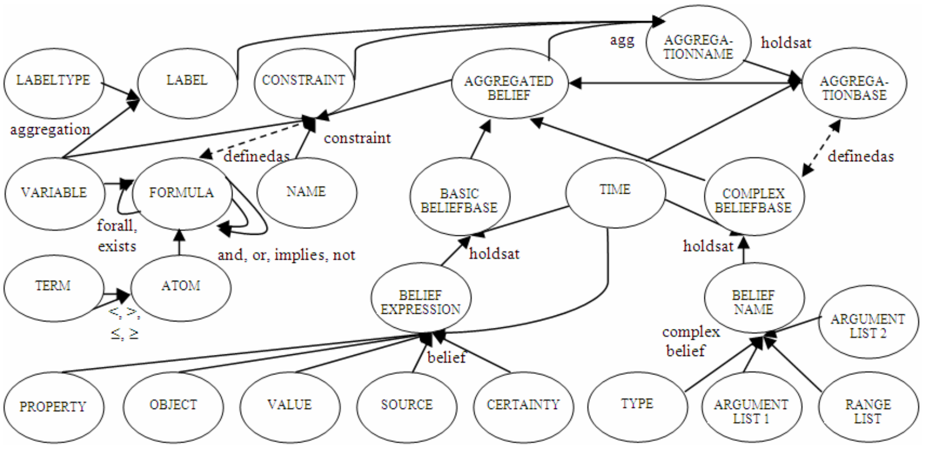


Fig. 1. Overview of the algebra for belief aggregation

A number of sorts are considered primitive; they only contain constants such as names and values: LABELTYPE, VARIABLE, PROPERTY, OBJECT, VALUE, TIME, SOURCE, CERTAINTY, RANGE, and TYPE. Some other sorts are more or less standard, and/or may depend on application dependent functions: ATOM, TERM, FORMULA.

Sort ARGUMENTLIST 1 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist; *free* and *range*, which denote that the argument of that position is variable. The sort RANGELIST contains terms listing the 6 ranges for the 6 arguments of ARGUMENTLIST 1. The range is only relevant for the arguments with the special instance *range*. In the case of a normal instance the corresponding range is *nr* (not relevant) while in the case of the special instance *free*, the corresponding term is *any*. Sort ARGUMENTLIST 2 contains terms listing 6 arguments with at each of the 6 positions instances. Two special instances exist; *given* and *nr*, which denote respectively that the argument of that position was already specified in ARGUMENTLIST 1, or is no longer relevant given the TYPE.

Note that the function agg can be used in a recursive manner together with the function holdsat. The nested term structures that result, represent beliefs at different levels of aggregation: the level is the number of nested agg functions occurring in the term.

The area of algebraic specification has a long history. From the extensive literature techniques can be borrowed to obtain an implementation of calculations in the algebra, for example in a functional or logic programming language. If relations are involved, an implementation has to take into account both functional and logical aspects; e.g., [5, 6]. Following this tradition, the next section introduces an implementation of the developed algebra in the logic programming language Prolog.

The algebra is considered a term algebra, which specifies the different variations of aggregated belief expressions that can be formed. The current Prolog implementation generates such expressions, but does not perform evaluations of whether two different expressions should be considered as having the same content or meaning. In future work it is planned to extend this approach to an algebra for which also equations are specified, and an implementation where such equations are incorporated.

5 Implementation

The algebra of section 4 is implemented in SWI-Prolog [7]. In this section, the implementation choices are explained. For the readability of this section, only parts of the Prolog program are shown. The complete source code can be downloaded from: http://www.few.vu.nl/~heuvel/CIA-AggregationAlgebra.pl

5.1 Controlling Aggregations

The current implementation does not incorporate automatic control of aggregations. Instead two ‘programs’ are implemented that can be called from the Prolog-shell: *holds* and *post*. The *holds* program is shown below and can be used to request the results of a specific aggregation, or to request the values for which a specific complex belief holds. Notice that complex beliefs are defined as aggregations and are as such interchangeable. When no holds_at attribute is included in the query, it is assumed that the query requests the result of the aggregation or complex belief at the current time. When a holds_at is included, the query requests the result of the aggregation or complex belief that holds at the specified time.

```
holds(B):-
     B = complex_belief(_, _, _, _),
     current_time(N),
     complex_belief_is_defined_as(
        holds_at(B,N),
        holds_at(agg(L,C,A),N)),
     holds_at(agg(L,C,A),N).
```

Query about B,
B is a complex belief, and is checked for the current time N.
The definition of the
complex belief B is
the aggregation agg(L, C, A),
which is requested for time N.

```
holds(B):-
     B = holds_at(X,N),
     X = complex_belief(_, _, _, _),
     complex_belief_is_defined_as(
        B, holds_at(agg(L,C,A),N)),
```

Query about B,
B is whether X holds at time N,
with X being a complex belief.
The complex belief X within B is defined as an aggregation.

`is_time(N),`
`holds_at(agg(L,C,A),N).`
	When N is an actual time, that aggregation is requested for N.

`holds(B):-`
`B = agg(L,C,A),`
`current_time(N),`
`holds_at(agg(L,C,A),N).`
	Query about B,
	B is an aggregation, and is checked for the current time N, and therefore requested at N.

`holds(B):-`
`B = holds_at(agg(L,C,A),N),`
`is_time(N),`
`holds_at(agg(L,C,A),N).`
	Query about B,
	B is whether agg(L,C,A) holds at N, when this is an actual time agg(L,C,A) is requested at N.

The *holds* program does not alter the belief-base and as such can be used to investigate 'what-if' questions. Besides the *holds* program also a *post* program exists whose procedure is almost identical, except each of its rules is extended with the extra condition *assert(_)*. This adds the complex belief, defined as the aggregation that is checked to hold at N, to the belief base.

5.2 Free and Bounded Variables

Complex beliefs do not have 6 atomic arguments like basic beliefs, but are made up of four arguments. The first of these is atomic and specifies a name for the complex belief, which is also referred to as its *type*. The latter three arguments are predicates; each embeds 6 arguments whose positions respectively represent the P, O, V, T, S, and C. So a complex belief is represented by *complex_belief(Type, For(…), With_Ranges(…), Has_Values(…))*, which denotes that it is believed by the agent that for that complex_belief *Type* and for the given constants in *For*, taken into account the *With_Ranges* in which the free variables in *For* have to lie, the constants in *Has_Values* count.

An example complex belief denoting that it is believed that the last time at which a belief was held about the hostile identity of contact1 is 7, and that was with a certainty 0.6 and based on radio contact is: *complex_belief (last, for(identity, contact1, hostile, free, free, free), with_ranges(nr, nr, nr, any, any, any), counts(given, given, given, 7, radio, 0.6))*. Such a complex belief is the result of the agent reasoning about what the last time, i.e. highest T, was that it believed that the identity of contact1 was hostile. When it would have reasoned about what the last time was that it believed with less than 0.5 certainty that that was the case, the following complex belief might have hold: *complex_belief (last, for(identity, contact1, hostile, free, free, range), with_ranges(nr, nr, nr, any, any, [0, 0.5]), has_values(given, given, given, 4, vision, 0.4))*.

Given this representation of complex beliefs, an example of a complex_belief_is_defined_as relation which defines a complex belief as a specific aggregation is:

```
complex_belief_is_defined_as(
   holds_at( complex_belief(
      last,
      for(P,O,V,free,S,C),
      with_ranges(nr,nr,nr,any,nr,nr),
      has_values(given,given,given,X,given,given)),N),
holds_at( agg(temporal_aggregation(T),
              highest_free(X,any,P,O,V,S,C),
              holds_at(belief(P,O,V,T,S,C),N)),N)).
```

The aggregation shown here will return the highest T that it can find for the given P, O, V, S, and C. When it does not matter what the S and C are, but it is required to find the highest (last) T that is now restrained to a certain time range [Tb, Te] for a given P, O, and V, the following aggregation is applicable:

```
complex_belief_is_defined_as(
   holds_at(   complex_belief(
      last,
      for(P,O,V,range,free,free),
      with_ranges(nr,nr,nr,[Tb,Te],any, any),
      has_values(given,given,given,X,Y,Z)),N),
   holds_at(   agg(
      temporal_source_certainty_aggregation(T,S,C),
      highest_range_free_free
         (X,[Tb,Te],Y,any,Z,any,P,O,V),
      holds_at(belief(P,O,V,T,S,C),N)),N)).
```

This aggregation will return the highest T that it can find for the given P, O, and V. The S and C that it returns are those of the belief with that highest T. This aggregation example demonstrates that variables can be free or that they can be restricted to a specific range. When it is checked whether a certain aggregation holds at a certain time the following clause executes:

```
holds_at(agg(L,C,A),_):-
     term_variables(L,V),
     constraint_is_defined_as
        (constraint(C,A,V),F),
     F.
```

To determine the agg(L, C, A) at time _,
the variables in L are listed in V.
The definition of the constraint C for
the subject A and the variables in V is F,
which is consequently requested.

The first condition term_variables(L,V) is a built-in Prolog predicate that unifies V with a list of variables, each corresponding with a unique variable of L and ordered in order of appearance in L. So for the example above it holds:

```
?- term_variables( temporal_certainty_source_aggregation(T,C,S),V).
V=[G34,G35,G27], T=G34, C=G35, S=G27.
```

The second condition is a user-defined predicate that defines what the constraint C entails for the aggregated belief A with its free variables listed in V; namely F, which forms the last condition. On the next page, an example constraint_is_defined_as is shown for the constraint that is required to deduce the first complex belief of type *last* introduced in this section. Notice that this *highest_free* constraint can be reused, e.g., to deduce a complex belief of type *surest* when it is combined with a certainty_aggregation. Its logical expression is:

given A,

$$\forall x \quad [\, highest\_free\,(x, any) \leftrightarrow A(x) \wedge \forall x1\; [A(x1) \rightarrow x1 \leq x\,]\,] \tag{3}$$

```
constraint_is_defined_as(                           | Definition   (
constraint(                                         | constraint (
  highest_free(X,any,F1,F2,F3,F4,F5),               |   C,
  A1, [X1]),                                        |   A, V    ),
and( copy_term( (A1, X1, F1, F2, F3, F4, F5),       |
                (A, X, F1, F2, F3, F4, F5)),        | F            ).
    and(A, forall(A1, X1 =< X)))).                  |
```

The constraint that was required to deduce the second complex belief of type *last* introduced in this section, is shown next. It can be seen that this constraint only considers options A1 whose values X1 for the variable X lies within the range [Xb, Xe] specified for it.

```
constraint_is_defined_as(                           | Definition   (
constraint(                                         | constraint (
  highest_range_free_free(                          |
     X,[Xb,Xe],Y,any,Z,any,F1,F2,F3),               |   C,
  A1, [X1, Y1, Z1]),                                |   A, V    ),
and( copy_term( (A1,X1,Y1,Z1,F1,F2,F3),             |
                (A,X,Y,Z,F1,F2,F3)),                | F
  and(A, and( X>=Xb, and( X<Xe,                     |
    forall( and(A1, and( X1>=Xb, X1<Xe))            |
       X1 =< X)))))).                               |              ).
```

5.3 Nested Aggregations

The reason that in the constraint_is_defined_as Prolog clauses the values F1, ..., Fn are embedded is that although they are usually instantiated, they do not have to be. When they are not, and are left out of the query, they do get instantiated when Prolog requests A. However, when next is asked whether for all X1 in A1 X1 ≤ X holds, this probably fails. This is because the left-out variable that now is instantiated in A, is still free in A1, so much more A1's are checked than there should be.

The reason why variables are allowed to exist in places where atoms are expected is because this freedom enables nested aggregations. An example of a nested aggregation is the complex belief *integrated_sources* introduced in section 3:

```
complex_belief_is_defined_as(
   holds_at(complex_belief(
              integrated_sources,
              for(P,O,free,free,free,free),
              with_ranges(nr,nr,any,any,any,any),
              has_values(given,given,X,nr,nr,Y)),N),
   holds_at(agg(
              certainty_temporal_source_value_
                aggregation(C,T,S,V),
              highest_free_after_free_for_free_
                free_for_predicate_and_time
                Y,any,_,any,_,any,X,any,P,O,N),
              holds_at(complex_belief(
                last,
                for(P,O,free,free,S,free),
                with_ranges(nr,nr,any,any,nr,any),
                has_values(given,given,V,T,given,C)),N))
         ,N) )
```

In this clause a complex belief of type *last* functions as aggregated belief for the aggregation that deduces the complex belief of type *integrated_sources*. This latter aggregation aggregates over values, times, sources, and certainties of beliefs about a given property and object, in order to retrieve a specific value and certainty. The aggregation belief it needs as input is a complex belief of type *last* that aggregates over values, times and certainties for a given property, object and source. However, the latter (S) is not given but variable, because the top-aggregation needs this *last* type for all possible sources. Note that instead of the complex belief of type *last* also the aggregation as which it is defined could have been used as input.

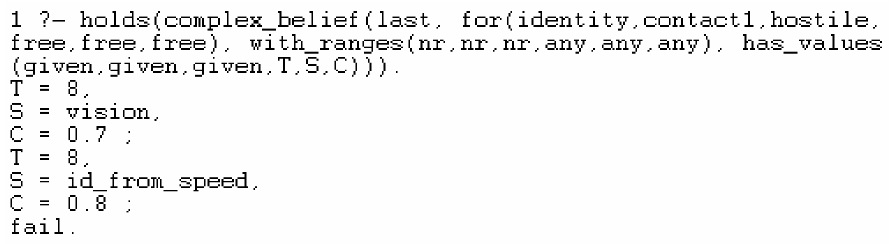
The constraint used within the aggregation to deduce the complex belief *integrated_sources* is much more specific and therefore less reusable than, e.g., the *highest_free* constraint. These two examples nicely illustrate the reach of the proposed aggregation mechanism. In principle all possible constraints can be added and used to form new types of complex beliefs that in turn can be used in other aggregations.

6 Example Scenarios

From http://www.few.vu.nl/~heuvel/CIA-AggregationAlgebra.pl the source code of our Prolog program can be downloaded. In the case presented, an agent attempts to infer the identity of a radar contact. Information about this contact can be gathered by the radar as by the agent's own vision. Furthermore, the agent can generate new beliefs by reasoning over other beliefs. Over time the following basic beliefs have held in STM and are now stored in LTM:

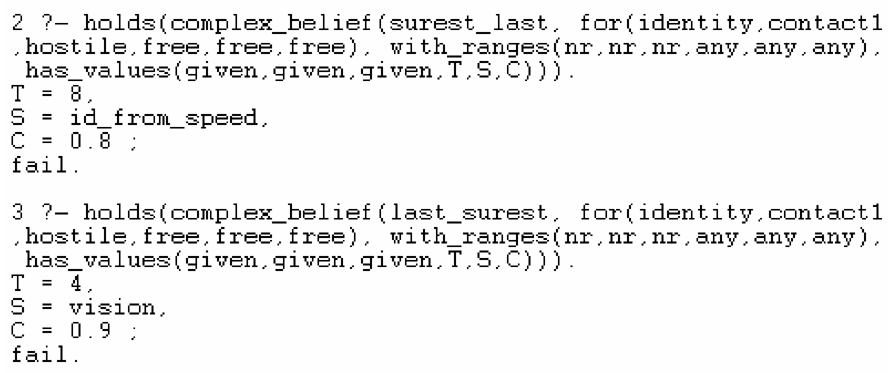
```
holds_at(belief(identity, contact1, neutral, 2,
                 vision, 0.5), 2).
holds_at(belief(identity, contact1, neutral, 3,
                 radar, 0.3), 3).
holds_at(belief(speed, contact1, 20, 3,
                 radar, 0.9), 3).
holds_at(belief(identity, contact1, hostile, 4,
                 vision, 0.9), 4).
holds_at(belief(identity, contact1, hostile, 4,
                 id_from_speed, 0.4), 4).
holds_at(belief(speed, contact1, 28, 6,
                 radar, 0.9), 6).
holds_at(belief(speed, contact1, 30, 7,
                 vision, 0.5), 7).
holds_at(belief(identity, contact1, hostile, 8,
                 vision, 0.7), 8).
holds_at(belief(identity, contact1, hostile, 8,
                 id_from_speed, 0.8), 8).
```

At current_time 10, two of the nine beliefs stored in the agent's LTM are formed by the agent's reasoning rule id_from_speed, which forms the source of those beliefs. At this moment the agent might start another reasoning process for which it requires the last belief about a hostile identity of contact1. This query results in:



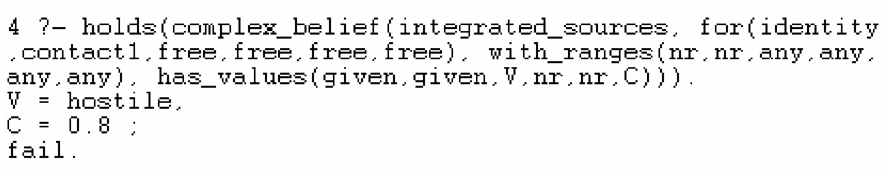
```
1 ?- holds(complex_belief(last, for(identity,contact1,hostile,
free,free,free), with_ranges(nr,nr,nr,any,any,any), has_values
(given,given,given,T,S,C))).
T = 8,
S = vision,
C = 0.7 ;
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.
```

By chance, two sets of atoms are found that both adhere to this query. In such case the agent might be interested in the surest one of these two last beliefs. This complex belief of type *surest_last* is formed by aggregating the label certainty_temporal_ source_aggregation and the highest_free_free_free constraint with that complex belief of type *last* as aggregated belief. The query for complex belief of type *last_surest* yields a totally different result: it is an aggregation of the same constraint but in combination with a temporal_certainty_source_aggregation and on complex beliefs of the type *surest*.



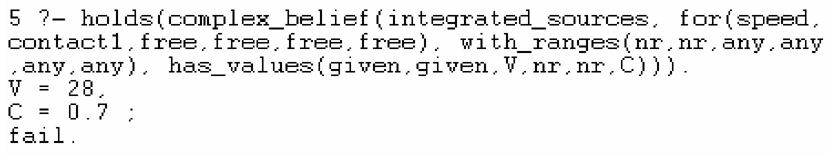
```
2 ?- holds(complex_belief(surest_last, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 8,
S = id_from_speed,
C = 0.8 ;
fail.

3 ?- holds(complex_belief(last_surest, for(identity,contact1
,hostile,free,free,free), with_ranges(nr,nr,nr,any,any,any),
 has_values(given,given,given,T,S,C))).
T = 4,
S = vision,
C = 0.9 ;
fail.
```

Another possibility would be that the agent's superior asks the agent what it believes that contact1's identity is. At that moment the agent will retrieve its last beliefs about the identity of that contact and form an answer. In the current case the agent believes contact1 might be neutral based on what it saw of the vessel, as on the radar-emission-pattern it received from the contact. However, it also believes it might be hostile, due to its high speed. In order to give its superior an answer the agent has to form a belief about the contact's identity by integrating the retrieved last information about its identity from the different sources. Given that the persistence-decay of a contact's identity (see section 3) is 0, the agent reports it believes the contact to be hostile since it was most sure of that.



```
4 ?- holds(complex_belief(integrated_sources, for(identity
,contact1,free,free,free,free), with_ranges(nr,nr,any,any,
any,any), has_values(given,given,V,nr,nr,C))).
V = hostile,
C = 0.8 ;
fail.
```

The agent's superior could also have asked what the agent believes that the speed of contact1 is. Again the agent needs to integrate information from different sources and times. However, because the persistence-decay of speed is larger than 0, say 0.05, it also has to take into account how long ago it was that it believed that information. The answer it will give is 28, see below. This knowledge is deduced from the basic belief at time 6 that its speed was 28, but notice that the certainty with which it is believed has decayed; from 0.9 to 0.7. Moreover, a newer belief concerning the contact's speed existed. However, even though the predicate's certainty decreases over time, still the value of the older belief is believed because the certainty of the new belief was very low.



```
5 ?- holds(complex_belief(integrated_sources, for(speed,
contact1,free,free,free,free), with_ranges(nr,nr,any,any
,any,any), has_values(given,given,V,nr,nr,C))).
V = 28,
C = 0.7 ;
fail.
```

7 Related Research

The technique for pre-processing a knowledge base to derive intermediate conclusions that is presented in this paper is related to the area of knowledge compilation. Knowledge compilation is defined in [8] as "methods of processing off-line a knowledge base in such a way that the output of such a pre-processing can be used to speed up on-line answering for a class of queries, where the pre-processing should take an finite amount of time". Within the area of knowledge compilation a distinction is made between exact methods (which are sound and complete) and approximate methods, which either reduce the complexity by expressing the knowledge or query in a simpler language or by leaving out some (complex) parts of the knowledge base.

Our approach is an exact technique, as it only results in sound intermediate results. However, a difference with common techniques for knowledge compilation is that our method does not strive to derive all intermediate results, whereas knowledge compilation techniques usually try to find a representation of all theorems of the initial knowledge base. For example, they transform a knowledge base to normal form and compute all implicants or implicates. In contrast, our approach is driven by specific queries whose results are likely to be useful for the task execution. In that sense, our method is not complete, as it does not aim to represent all knowledge in a different representation. Moreover, our aggregations are usually more complex (and thus richer in information), whereas knowledge compilation techniques often result in simpler representations. Last, our aim is to compile new knowledge on-line instead of off-line.

Shahar [9] presents a framework for knowledge based temporal abstraction from time-stamped data. His formal specification of a domain's temporal-abstraction knowledge supports acquisition, maintenance, reuse, and sharing of that knowledge. His aim is partly the same as our, however, his framework allows for temporal abstractions only, whereas our algebra allows for arbitrary abstractions.

The area of belief revision is also related to our work. In belief revision, the question is how existing beliefs are influenced when new pieces of information are taken into account, for example when information is added, removed or changed. The dominant theory on belief revision, the so-called AGM model [10], formulates properties that an operator that performs revision should satisfy in order for being considered rational. Similarly, related work on belief merging focuses on the consequences of combining belief bases for the integrity of a belief base, for example, see [11]. In our work the logical consequences of the aggregations are not relevant, as no new knowledge is added. There is no inconsistent information that is merged and it does not happen that old information changes because all information is time-stamped. This is comparable to what Sripada [12] describes, who also uses time-stamped beliefs. He proposes a technique for the efficient revision of beliefs in knowledge bases for real-time applications, but only looked at binary beliefs.

Another type of related work is formed by approaches for memory storage in existing (cognitive) agent architectures. In a recent review study on computer-based human behaviour representations [13] it was generalized that "all the (human behaviour) models can represent either short term memory (STM) or long term memory (LTM)." However, the ways in which these memories function differ greatly. For example, ACT-R's STM is formed by a retrieval buffer that can hold one chunk, which it retrieves using an activation function from its declarative memory module (LTM) [14], while Soar's STM is formed by its working memory that is not limited in the number of elements it can hold [15]. Related to the differences in memories, differences exist in the representation of the declarative information entities stored in such modules. These representations range from nodes in a network with an activation value to first-order propositions.

The functioning of the various memories are in general fixed and tuned to bring about the behaviour for which the architecture was developed. No existing architecture is build to specifically deal with time-labelled constructs, let alone in the algebraic approach as introduced in this paper. Despite this, it might be possible to map the specific belief construct to the memory construct of an architecture, prohibited the form of the latter has a certain degree of freedom [2, 16]. Moreover, the constraints that are needed to infer required (possibly domain-specific) aggregations have to be implemented in the architecture as well, as the aggregation algebra.

8 Summary and Future Research

In this paper a method and a term algebra is presented to form arbitrary aggregations of beliefs in a knowledge base. The aggregations can be formed at different levels and from different perspectives, i.e. time aggregations, source aggregations, certainty aggregations, etc. A Prolog program is used to illustrate the feasibility of the approach. The motivation of this work is twofold: it should help to improve the computational problems when reasoning over a knowledge base, and it should reflect a more human way of storing information in memory. As such, the goal is to 'validly' represent aggregations of humans over beliefs, both conscious as subconscious, which can be used in agent applications where agents should behave in a human-like way.

Up to now, the control of the formation of aggregations is not yet implemented. Future research will investigate the control of aggregations from two perspectives. The first will be inspired by the human processes of forgetting and remembering, the second by the human processes of attention and focusing in task execution.

References

1. Heuvelink, A., Both, F.B.: A Cognitive Tactical Picture Compilation Agent. In: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007), pp. 175–181. IEEE Computer Society Press, Los Alamitos (2007)
2. Both, F., Heuvelink, A.: From a Formal Cognitive Task Model to an Implemented ACT-R Model. In: Proceedings of the 8th International Conference on Cognitive Modeling (ICCM 2007), pp. 199–204. Psychology Press (2007)
3. Castelfranchi, C.: Representation and Integration of Multiple Knowledge Sources: Issue and Questions. In: Cantoni, V., Di Gesù, V., Setti, A., Tegolo, D. (eds.) Human & Machine Perception: Information Fusion, pp. 235–254. Plenum Press (1997)
4. Bloch, I., Hunter, A., et al.: Fusion: General Concepts and Characteristics. International Journal of Intelligent Systems 16(10), 1107–1134 (2001)
5. Drosten, K.: Translating Algebraic Specifications to Prolog Programs: a Comparative Study. In: Algebraic and Logic Programming. LNCS, vol. 343, pp. 137–146. Springer, Heidelberg (1988)
6. Hanus, M.: The Integration of Functions into Logic Programming: From Theory to Practice. Journal of Logic Programming 19, 20, 583–628 (1994)
7. Wielemaker, J.: An Overview of the {SWI-Prolog} Programming Environment. In: Proceedings of the 13th International Workshop on Logic Programming Environments, pp. 1–16 (2003)
8. Cadoli, M., Donini, F.M.: A Survey on Knowledge Compilation. AI Communications 10(3-4), 137–150 (1997)
9. Shahar, Y.: A Framework for Knowledge-based Temporal Abstraction. Artificial Intelligence 90(11), 79–133 (1997)
10. Konieczny, S., Pino Pérez, R.: Merging Information Under Constraints: A Logical Framework. Journal of Logic and Computation 12(5), 773–808 (2002)
11. Alchourròn, C.E., Gärdenfors, P., Makinson, D.: On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. Journal of Symbolic Logic 50, 510–530 (1985)
12. Sripada, S.M.: A Temporal Approach to Belief Revision in Knowledge Bases. In: Proceedings of the Ninth Conference on Artificial Intelligence for Applications, pp. 56–62. IEEE Computer Society Press, Los Alamitos (1993)
13. Morrison, J.E.: A Review of Computer-Based Human Behavior Representations and Their Relation to Military Simulations. Institute for Defense Analyses, Paper P-3845 (2003)
14. Anderson, J.R., Lebiere, C.: The Atomic Components of Thought. Lawrence Erlbaum Associates, Mahwah (1998)
15. Laird, J.E., Newell, A., Rosenbloom, P.S.: SOAR: An Architecture for General Intelligence. Artificial Intelligence 33(1), 1–64 (1987)
16. Muller, T.J., Heuvelink, A., Both, F.: Comparison of Implementations of a Cognitive Model in Soar and ACT-R. In: Proceedings of the 6th International Workshop on From Agent Theory to Agent Implementation (AT2AI-6) (2008)

Software Engineering for Service-Oriented MAS

Emilia Garcia, Adriana Giret, and Vicente Botti

Department of Information Systems and Computation, Technical University of Valencia, Camino de Vera, Valencia, Spain
{mgarcia,agiret,vbotti}@dsic.upv.es

Abstract. Nowadays, service-oriented architectures (SOA) and multiagent systems (MAS) are two increasingly important technologies. Despite the differences in technology, SOA and MAS have some similar objectives and their integration produces systems with more flexibility, functionality and interoperability. Their integration creates new requirements and special methods and tools are necessary to develop systems that integrate both technologies. This paper analyzes the most important issues for developing Service-oriented MAS. Furthermore, some methods and tools to develop this kind of systems are analyzed to show how current approaches solve the problem of the integration between agents and services.

Keywords: Multiagent systems, service-oriented architectures, software engineering, development tools.

1 Introduction

Nowadays, SOA and MAS are two increasingly important technologies. The objectives of both architectures share some similarities, i.e., both of them try to create distributed and flexible systems that are composed of loosely-coupled entities which interact with each other.

Despite these similarities, there are major differences in their technology. Services have interface standards and exchange protocols that are completely different from agent communication languages and protocols. This is why they cannot interact with each other directly.

This is a problem which needs to be solved, but some studies [15] have investigated this issue and show that the integration of agents and services produces attractive benefits. Services have a well-defined infrastructure and interoperability whereas agent technology aims to provide intelligent and social capabilities (trust, reputation, engagement, etc) for applications. Therefore, the integration of agents and services improves the flexibility, interoperability and functionality of the system.

Nevertheless, most agent software engineering techniques do not consider integration with services, nor do service software engineering techniques consider integration with agents.

There are some works that address the integration between agents and services [8]. They define frameworks and provide tools for developing systems in which agents and services are integrated. Each of them has its own integration mechanism, communication language, and even its own service and agent concepts.

In order to define the software engineering issues for developing Service-oriented MAS, a detailed study of the state of the art of software engineering for agents, services and service-oriented agent systems is made and briefly summarized in Section 2. Section 3 defines a list of the most important software engineering requirements for developing Service-oriented MAS. Furthermore, some software engineering tools for developing systems that integrate agents and services have been analyzed to define how current approaches and tools solve the problem of the integration between agents and services. A description and a brief analysis of some selected tools and frameworks is presented in Section 4. Finally, Section 5 presents some conclusions and future work.

2 Background

This section is divided in three parts. Firstly, the state of the art of software engineering for MAS is briefly described. Secondly, the state of the art of software engineering for SOA is summarized. Finally, the state of the art of software engineering for service-oriented agent systems is analyzed.

2.1 Agent-Oriented Software Engineering

MAS are complex systems with a distributed nature, where each element is autonomous, reactive, proactive and social. This complexity makes the use of techniques and tools to support the development process necessary. Agent-oriented software engineering is based on traditional software engineering, but it takes into account the specific features of the agent technology. On the market today there are a great number of methodologies, development environments, modeling languages, debugging tools and platforms that deal with the development process of a multiagent system [16].

Some well-known tools and methodologies include AUML [21], Jade [1] , JACK [9], Gaia [23], Tropos [7].

The two main drawbacks to software engineering tools of this kind are the gap between modeling and platforms, and the lack of automatic and complete translation between the models and the executable code [18].

Agents usually use specific technology to represent ontologies, protocols and content languages. This makes the interaction with other types of systems difficult and sometimes necessitates the use of intermediary elements.

2.2 Service-Oriented System Engineering

It is based on traditional system engineering, but it must take into account specific characteristics of SOA systems. System engineering needs to be collaborative because SOA applications are often collaborative. Service consumers, service brokers and service providers collaborate to invoke, search, register and provide services. Systems may be composed at runtime using existing services so many SOA engineering tasks need to be done on the fly at runtime.

Services are oriented to being reused, so it is very interesting for them to have platform-independent modeling techniques.

Recently some initial results have been proposed [20,12]. These works have mainly concentrated on developing a methodology for service-oriented engineering and design-time models. In the first line of research, works have concentrated on how to provide sufficient principles and guidelines to specify, construct, refine, and customize highly volatile business processes choreographed from a set of internal and external Web services [22,13]. In the second line of research, some works have concentrated on developing design-time models using goal-oriented requirement analysis techniques [17,14].

2.3 Service-Oriented MAS Engineering

Agents and services have clear differences, such as the different representational encodings and technologies, for example. Despite these differences though, they have similar goals. Both technologies build flexible and distributed systems. Some researchers have studied the benefits of mixing these technologies [15]. They state that using agents and services can provide systems with more flexibility, functionality and interoperability. Many works on this subject are focused on the interaction mechanism between agents and services [11], but there is no work that provides a complete description of which are the software engineering requirements for developing this kind of systems. A detailed list of the software engineering features to take into account in the Service-oriented MAS development is presented in Section 3. Furthermore, there are few development environments and tools that actually offer facilities for developing agents that interact with services. Some of these tools are described and analyzed in Section 4.

3 Software Engineering Requirements

The aim of this section is to make an overall analysis of the needs that arise when developing Service-oriented MAS, relying more on the concepts than in a specific terminology. In the specialized literature, there are a great number of different perspectives about the integration between agents and services; there are even different conceptions of what an agent and a service are. Because of this, the identification of a set of independent, orthogonal features which completely characterize the Service-oriented MAS development process seems unfeasible.

The selection of these issues is based on the background briefly described in Section 2 and the study of current frameworks and techniques for Service-oriented MAS engineering that is summarized in Section 4.

These issues are classified into four categories: (1) Integration between agents and services; (2) Development issues; (3) Multiagent systems; (4) Service-oriented architectures.

3.1 Integration between Agents and Services

Firstly, it is important to analyze the way in which the relationship between agents and services is considered. Drawing from the published literature and, as described in [5], there are three different ways: (1) Some approaches hold that there is no conceptual distinction between agents and services. For them, both are active building blocks in a loosely-coupled architecture, and there is only an engineering problem of creating overall systems behaviors from active components [24]. (2) Other approaches hold that agents and services can communicate in a bidirectional way. They have to provide a mapping between the language protocol used in the multiagent system and the service language protocol and vice versa [2,25]. (3) Finally, other approaches hold that the communication is only useful in one direction, i.e., agents invoke services but not vice versa. In this view, agents are responsible for the application, and they use services or composite services as resources to achieve their objectives [4].

Furthermore, agents and services have different **communication standards**. Agents usually use FIPA ACL messages. Services are described with WSDL descriptions and use SOAP as the communication mechanism. However, not all the approaches follow the standards. The use of different technologies in services and agents and the **mechanism to match** them should be evaluated. Another point to consider is if the approach offers the possibility of dynamically **publishing and discovering services**. Some approaches even discuss the possibility of interacting with **external services** that may or may not be registered in the system.

3.2 Development Issues

This section analyzes the general features that a method and a development environment for developing Service-oriented MAS should have. These features are extracted from traditional software engineering and are grouped into three categories:

3.2.1 Software Engineering Support
As described in Section 2, the development of a system that integrates agents and services is a complex task which greatly benefits from the adoption of software engineering techniques. Firstly, the **application domain** should be considered. Some approaches are supposed to address systems from any domain and other are oriented to specific domains. The **model-central element** is another key feature; it defines the initial point and the perspective of the modeling process. Offering a **methodology** that involves agents and services and that also take into account their interaction greatly helps the developer to go from the initial information to the final implementation. There are lots of methodology features that can be analyzed, but the most important are which parts of the development process are covered and whether development guidelines are provided. Furthermore, a methodology component that can process a user behavioral description of desired functionality and recommends that the behaviour should be implemented via a service or an agent, is a very useful feature for developing

this kind of systems. The **modeling language** specifies the type of notation used for modeling. It can be formal or informal and may or may not use graphical elements. The notation should be precise, complete and clear. The systems are usually very complex, so it is very useful if the language modeling allows to model at different abstraction level. A very interesting feature is to offer mechanisms to specify semantics of model element extensions using formal methods, such as OCL.

3.2.2 Technical Issues
There are many approaches that only analyze the development process in a theoretical way. They define methods, but they do not offer **tools** or development environments to model, design and implement systems. Thus, one evaluation criterion is whether the approach offers tools and which parts of the **development process** are covered by them. These tools can be evaluated based on many criteria, but the most important are the requirements to set up and run it, the functionality offered, the ease of use and their scalability.

3.2.3 Implementation Issues
The most advanced development environments integrate the modeling, design and implementation processes in the same tool. They even offer tools that **automatically generate parts of the code** from the models. These characteristics are very desirable because they reduce the implementation time and the number of implementation errors. In the development process of systems that integrate agents and services, the **translation** from one descriptive language to another is very useful. This requires that the development environment has to offer a mechanism to automatically translate from one standard to another.

3.3 Multiagent Systems

There are many works that analyze the features that should provide the methodologies and the development environments for multiagent systems [6,18]. They describe a great number of criteria, but, in this paper, only the most important criteria for the specific case where agents interact with services are considered.

The multiagent system approach should have an **agent architecture** that carry out the fundamental **properties** of agents (autonomy, reactivity, sociality and proactiveness). The **content language** used for the communication mechanism and whether or not the architecture is FIPA compliant are very important features because they are strongly related with the necessary mechanism to interact with and to integrate services and agents.

3.4 Service-Oriented Architectures

As explained in Section 2.2, there are some features that should be taken into account for developing systems that use service-oriented technology.

The service architecture should be modeled **independently of a specific platform** to obtain more flexibility and reusability. The **platform-dependent**

characteristic also need to be specified to implement the system. Service-oriented systems are usually composed of many **standards** (See 2.2). Therefore, the development environment should provide facilities to implement and check the correctness of these standards. Another feature to take into account is the way **service descriptions** are implemented. Service specification provides a means for defining complete service specifications that include behavioral rules in addition to static interfaces, operations, preconditions, post conditions, and constraints. Service specifications are not architecturally neutral. Services can be **composed** to achieve more complex functionality. The mechanism to specify how services are composed from other services in the models and how the composition is translated to an executable code is another important feature.

4 Frameworks and Techniques

A brief description and analysis of some frameworks and techniques for modeling, designing and implementing systems of this kind is given below. This selection was made due to the relevance of the researchers and companies responsible for these tools and the fact that they cover a range of pertinent issues and supporting technologies that are primarily focused on the integration of agents and services.

– **The Nuin agent platform.** Nuin [4][5] is an open-source Java implementation which is a combination of a belief-desire-intention (BDI) agent platform and semantic web techniques. It provides an abstract service boundary to add custom behaviours to the agent. This abstract service boundary also provides a natural basis for extending the internal agent services in order to include external web services.
– **JASE.** It [3] is a Java-based Agent-oriented and Service-oriented Environment for deploying dynamic distributed systems. It defines a service-agent programming model, which is a combination of two concepts in the field of distributed computing: the concept of services and the concept of mobile agents. In JASE, mobile agents are used to support applications, and service interface agents are used to wrap services.
– **The Agent Modeling Language (AML).** AML [19] is a semi-formal visual modeling language for specifying, modeling and documenting systems that incorporate features drawn from multiagent systems theory. AML also supports the modeling of services and their interaction with agents.
– **The framework for Rapid Prototyping of SOA.** This framework is proposed by Zinnikus in [25]. It is built around a Model-Driven Development methodology that is used for transforming high-level specifications of SOA into executable artefacts, both for Web Services (WSDL files) and for BDI agents. It follows the OMG Model-Driven Architecture (MDA) approach and defines a Platform-Independent Model (PIM) for SOA (PIM4SOA) and Platform-Specific Models (PSMs) for describing Web services (XSD and WSDL), JACK BDI agents and BPEL processes. This framework is composed of three parts: a modeling part, a service part and an autonomous

agent part. The modeling part is concerned with applying Model-Driven Development (MDD) techniques and tools to the design of SOAs. It defines models and transformations that are specific to the concepts used for SOAs, such as Web Service descriptions and plans for autonomous agents. The service part provides a highly flexible communication platform for Web services. The autonomous agent part deals with designing and enacting service compositions as well as performing mediation, negotiation and brokering in both SOAs.
– **Jade web services integration gateway (WSIG).** WSIG [2] is a Jade add-on that provides support for bidirectional invocation of Web services from Jade agents, and Jade agent services from Web services clients.

4.1 Tools Analysis

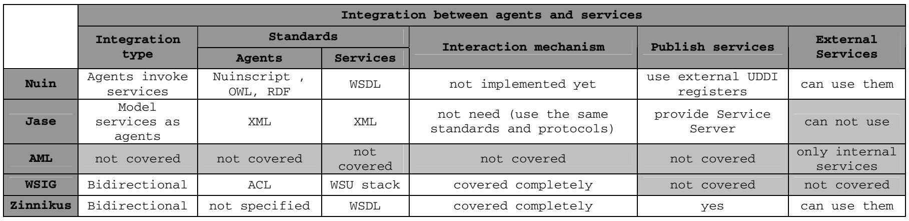
In this section, some of the features of these frameworks are related to show how current approaches develop this kind of systems, but the goal of this section is not to make an extensive and complete analysis of these approaches. Figures 1, 2 and 3 summarize this analysis highlighting the parts that are not covered by the analyzed tool.

4.2 Integration between Agents and Services

The Nuin approach considers that interaction between agents and services is only useful in one direction. Agents primarily are responsible for mediating between user goals and the available strategies and plans. Agents invoke atomic or composite external web services as necessary. An initial approach at integration of semantic web capabilities is to include the use of RDF/OWL as a knowledge representation, and the ability to use RDQL to query RDF-based knowledge stores. A future goal for the Nuin approach is to add support for the direct use of semantic web service descriptions in OWL-S. Nuin assumes that an appropriate binding to the abstract service is defined, but the interaction between Nuin agents and web services has not yet been implemented (see Figure 1).

In JASE, the general idea of service is that the application is separate from the resources needed to fulfill a task; these resources are modeled by services, which are independent of the application. JASE models services as agents. A service interface agent encapsulates a local resource. Each service interface agent consists of two parts: a service agent and a service interface. A service interface acts as a front-end interface for the other agents in the system to communicate with the service agent it represents. A service agent is a specialized service, which can be realized in the form of software or hardware. JASE uses XML to describe both service descriptions and agent queries, so no gateway is necessary.

In AML, services are encapsulated blocks of functionality that the entities can offer to perform upon request. AML is only a modeling language, so it represents a bidirectional interaction between agents and services. However it does not consider the technological differences between agents and services. The services publication and their discovery are not considered, either.



	Integration between agents and services					
	Integration type	Standards		Interaction mechanism	Publish services	External Services
		Agents	Services			
Nuin	Agents invoke services	Nuinscript , OWL, RDF	WSDL	not implemented yet	use external UDDI registers	can use them
Jase	Model services as agents	XML	XML	not need (use the same standards and protocols)	provide Service Server	can not use
AML	not covered	not covered	not covered	not covered	not covered	only internal services
WSIG	Bidirectional	ACL	WSU stack	covered completely	not covered	not covered
Zinnikus	Bidirectional	not specified	WSDL	covered completely	yes	can use them

Fig. 1. Integration issues

WSIG is a gateway that offers automatic, bidirectional operation allowing both FIPA compliant agent services and Web services to be registered with it. Agent services and web services can thereby publish their service descriptions to consumers outside their normal operational domain. The gateway can then intercept calls to these registered services allowing agents to invoke Web services and vice versa by transforming message encodings and creating service access endpoints. All invocation-related interactions between the gateway and agents use ACL encoded FIPA-Request and FIPAInform performatives. All Web services use the standard WSU stack (WSDL, SOAP and UDDI).

The Zinnikus approach extends the JACK agent framework for Web Services in order to provide a goal-oriented service composition and execution module within a SOA. Following the MDA approach, at design time a modeller specifies a set of plans (PSM level) that constitute the workflow library of the agents. Web service calls are integrated as steps into plans. Service providers are mapped to JACK agents/teams. The parts of the PIM that define the processes involved are mapped to agent/team plans and correlated events, whereas the parts that define the interfaces are mapped to the modules that provide the client- and server-side code for the JACK agent platform. Johnson and Lyndon are tools of the service part of the framework and allow the communication between external and internal web services and agents. The Johnson tool is responsible for invoking web services and receiving calls issued by Web service clients. The Lyndon tool takes WSDL files as input and configures Johnson tool to play either the role of service provider, service consumer or service proxy for the service described by the WSDL file analyzed.

4.3 Development Issues

4.3.1 Software Engineering Support

In all the studied tools, the application domain is general as shown in Figure 2. Even though JASE models can describe any kinds of open and global dynamic distributed systems, it is specialized in mobile agents.

The central element of the model in all cases is the agent, except for the Zinnikus approach. This approach defines first a platform-independent model for services (PIM4SOA) and later platform-specific models that have agents as central elements.

Nuin and JASE do not provide or use any methodology, nor any modeling language.

AML is a modeling language that is specified as an extension to UML 2.0 in accordance with major OMG modeling frameworks (MDA, MOF, UML, and OCL). It proposes 11 diagrams that extend UML 2.0, to model all the MAS features and interactions with services graphically. The notation is clear, complete, precise and understandable. AML offers the possibility to model at different abstraction level. ADEM is the agent methodology proposed for AML researchers but there is no public available detailed documentation of ADEM.

WSIG is a transparent gateway to translate communication standards, so it cannot be analyzed with these features.

As explained in Section 4, the Zinnikus approach is built around a Model-Driven Development methodology that transforms high-level specifications of a SOA into executable artefacts, both for web services (WSDL files) and for BDI agents. The modeling part of the framework and more specifically, the MDD framework defines the metamodels used to specify SOAs. It also provides modeling guidelines, model transformation and generation support for execution artefacts such as WSDL files and BDI plans. It also supports importing existing WSDL files into the SOA models. All these models are represented graphically.

4.3.2 Technical Issues

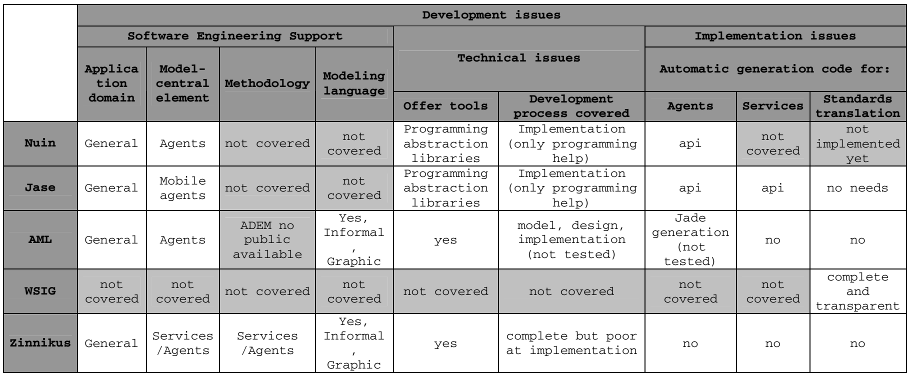
Nuin and JASE do not provide any development environment to implement applications. Nuin models core BDI agent architectures on AgentSpeak(L) and PRS, allowing agent designers to specify and implement agents using programming abstractions that correspond closely with the terms commonly used in intelligent agent theories. It is written in Java, and requires JDK 1.4 or later. Jade and Jena libraries are also necessary for a full functionality.

JASE is also implemented in Java. Similar to Nuin, JASE provides programming abstraction libraries.

AML is supported by tree case tools: Rational Rose 2003, Enterprise Architect 4.0 and StarUML. The AML implementation consists of UML profile support for AML, a set of modeling utilities (specialized element specification dialogs, model consistency checker, etc.), and forward-engineering tools for TAPI, the commercial-agent platform of Whitestein Technologies AG.

The WSIG requires JADE v3.3 platform to run, and the following third party technologies are available on the system: Jakarta Tomcat, Apache jUDDI, mySQL, MySQL Conector/J.

The Zinnikus approach provides tool support for the MDD framework. It has been developed as a set of plugins for Rational Software Modeller (RSM) (IBM Rational Software). RSM is a UML 2.0 compliant modeling tool from IBM based on the Eclipse modeling environment. All models and metamodels were implemented using the EMF Core (Ecore) metamodel. Model transformations have been implemented using the model transformation capabilities of the RSM/Eclipse platform. Also [25] provides Johnson tool, Lyndon tool, WSDL Analyzer (a tool for detecting similarities at a structural level between WSDL



	Development issues								
	Software Engineering Support				Technical issues		Implementation issues		
							Automatic generation code for:		
	Application domain	Model-central element	Methodology	Modeling language	Offer tools	Development process covered	Agents	Services	Standards translation
Nuin	General	Agents	not covered	not covered	Programming abstraction libraries	Implementation (only programming help)	api	not covered	not implemented yet
Jase	General	Mobile agents	not covered	not covered	Programming abstraction libraries	Implementation (only programming help)	api	api	no needs
AML	General	Agents	ADEM no public available	Yes, Informal , Graphic	yes	model, design, implementation (not tested)	Jade generation (not tested)	no	no
WSIG	not covered	not covered	not covered	not covered	not covered	not covered	not covered	not covered	complete and transparent
Zinnikus	General	Services /Agents	Services /Agents	Yes, Informal , Graphic	yes	complete but poor at implementation	no	no	no

Fig. 2. Analysis of the development issues

descriptions of Web services and generating the corresponding mappings) and RDF store (which stores both design-time information and runtime information as RDF files for the purpose of monitoring).

4.3.3 Implementation Issues

As explained in the above sections, JASE and Nuin do not offer any modeling mechanism, so the translation between models and code is not considered.

The CASE tools provided by AML are supposed to be able to translate agent models into Jade code [10], but these plugins are not publicly available.

The WSIG v0.4 supports the standard WSU stack and FIPA acl/sl0 communication. The functionality of WSIG is translated between these two standards. WSIG automatically registers UDDI web service registers in the Jade platform DF and viceversa, i.e., WSIG registers all agent services of the DF in the WSIG UDDI repository. WSIG is transparent, so the programmer does not need to add any code in agents or services. An agent calls a web service as to another agent and viceversa.

The Zinnikus approach provides model-to-model transformation services that allows the transformation of PIM4SOA models into underlying PSMs such as XSD, WSDL, JACK BDI agents or BPEL. However, it does not specify if JACK agents code is generated automatically from these models.

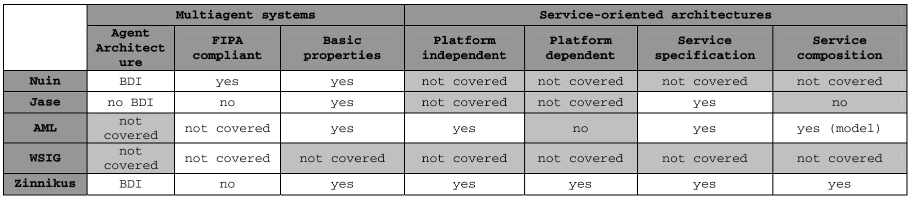
4.4 Multiagent Systems

Nuin agents are BDI agents that are reactive, autonomous, proactive and social.

JASE is not a BDI architecture. It provides a mechanism for developing agents with all the basic properties described in Section 3 as well as with the ability to migrate.

AML allows agents to be modeled with all the basic properties.

WSIG is a transparent gateway to translate communication standards, so it cannot be analyzed with these features (see Figure 3).



	Multiagent systems			Service-oriented architectures			
	Agent Architect ure	FIPA compliant	Basic properties	Platform independent	Platform dependent	Service specification	Service composition
Nuin	BDI	yes	yes	not covered	not covered	not covered	not covered
Jase	no BDI	no	yes	not covered	not covered	yes	no
AML	not covered	not covered	yes	yes	no	yes	yes (model)
WSIG	not covered	not covered	not covered	not covered	not covered	not covered	not covered
Zinnikus	BDI	no	yes	yes	yes	yes	yes

Fig. 3. Agents and services issues

The Zinnikus approach extends the JACK agent framework for Web Services (JACK4WS) following the BDI model. The agents have all the basic properties. JACK is not FIPA compliant. JACK agents are not bound to any specific agent communications language. Nothing prevents the adoption of high-level symbolic protocols such as KQML or FIPA Agent Communication Language (ACL).

4.5 Service-Oriented Architectures

The objective of Nuin is not to implement services but to implement agents that can invoke services. Therefore, it cannot be analyzed with these features (see Figure 3).

JASE uses XML to describe both service descriptions and the mobile agent's queries. A service in JASE is a mechanism to encapsulate a local resource. JASE does not consider service composition.

AML does not cover most operational semantics, which is often dependent on a specific execution model given by an applied theory or deployment; it offers platform-independent models. The AML support for modeling services comprises (1) the means for the specification of the functionality of a service and the way a service can be accessed (service specification and service protocol), (2) the means for the specification of what entities provide/use services (service provision, service usage, and serviced property), and (if applicable) by what means (serviced port). They are modeled in AML in terms of service specifications, service provisionings and service usages. AML supports OCL.

WSIG is a transparent gateway to translate communication standards, so it cannot be analyzed with these features.

In the Zinnikus approach, service providers are mapped to JACK agents/ teams; the processes involved are mapped to agent/team plans, and interfaces are mapped to the modules that provide the client- and server-side code for the JACK agent platform. Thus service specification and interaction points are well-defined. The service composition is analyzed and implemented at the agent level, i.e., a composition is a collaboration between agents that are service providers.

4.6 Discussion

Some authors [5] say that if services are able to invoke agents, this would violate the autonomy of the invoked agent, thereby turning the agent into just another service. From our perspective, it is important to differentiate between agents and services. We agree that it is useful for agents to be able to invoke services. However, we disagree with the idea that if a service can invoke an agent, the agent must expose pre-determined or deterministic behaviours. Nowadays, service technology offers the possibility to register and deregister services dynamically and there is no reason why an agent cannot change the behavior of its published services depending on its own goals and situation. For this reason, we think that bidirectional integration is more complete and useful.

Organizations like FIPA or OMG are working to establish standards, and most approaches follow them. It is very important to obtain open environments where services and agents implemented by different companies with different technologies can interact.

As discussed in Section 2, developing systems that integrate agents and services is a complex task and the use of methodologies is useful. Nonetheless, there are few methodologies that take into account both technologies. The Zinnikus approach presents a methodology in [25] which states that the methodology is complete and that guidelines are offered. However there is no more documentation and they do not offer public downloads.

Figure 2 shows that these approaches do not offer tools that cover the entire development process. Also, there are few techniques for automatic code generation and they are not sufficient. These are very important research lines that are still open.

The possibility of offering platform-independent model services is very useful for develop service-oriented multiagent systems. It allows high-level modeling where the technology used is not specified. Complete approaches for developing systems that integrate services and agents should offer this possibility as well as automatic transformations to platform-dependent models.

From the list of features presented in Section 3 and the analysis of the results of Section 4.1 we can summarize the most important requirements for developing Service-oriented MAS in the following list: (1) a methodology that involves agents and services, and takes into account their integration; (2) a modeling language that allows the definition of the specific characteristics of agents, and services as well as the specification of their integration; (3) a tool that covers the entire development process, i.e., one that supports the methodology and the modeling language used and offers implementation facilities such as automatic code generation; (4) a gateway that allows the interaction between agents and services despite the differences in their technology and standards, which should provide mechanisms for publishing and invoking services; (5) an agent platform that integrate both technologies transparently.

After this analysis it can be observed that there is currently no complete tool or framework that covers the entire development process of service-oriented multiagent systems. In order to develop systems of this kind, a software engineer has to merge a set of different methods, modeling language, tools, etc.

5 Conclusions and Future Work

Multiagent systems and service-oriented architectures are two approaches with similar goals but major differences in their technology. Both are hot research and industrial topics, and integration between agents and services is very beneficial as it generates more complete, flexible and interoperable systems with greater functionality.

The use of software engineering principles, methods and techniques in the entire development cycle of multiagent systems and service-oriented architectures is both interesting and necessary in many cases. In the same way, the development of systems that integrate the two technologies requires methodologies and development environments that take into account the specific characteristics of agents, services and their integration.

In this paper we have put forward a comprehensive list of the most important software engineering issues in the development of service-oriented MAS systems. These issues were defined, based on a detailed study of state of the art development methods and taking into account the new characteristics which arise when agents and services are integrated.

These issues are used to analyze several approaches. From this study we can conclude that in order to develop systems of this kind, it is currently necessary to merge a set of different methods and tools. There is no software engineering tool that covers the entire development process of these systems, and so this is an open line of research.

The highlighting of the fundamental development issues put forward in this study is an attempt to identify the new requirements imposed to the development process of Service-oriented MAS. Moreover, this requirement list can help to improve current state of the art methods, tools and platforms in order to develop these kinds of systems correctly.

At the same time, the ideas presented in this study can be used as a starting point from which to develop a complete framework for service-oriented MAS, in which agents and services are smoothly integrated, drawing on the advantages of both approaches.

We plan to continue along this line of work in the future, completing the list of requirement, evaluating state of the art methods, tools and platforms in order to define a ranking list of development tools for these kinds of systems.

Acknowledgements

This work is partially supported by the TIN2006-14630-C03-01, PAID-06-07/3191 projects and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

1. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE (Wiley Series in Agent Technology). John Wiley & Sons, Chichester (2007)
2. Board, J.: Jade web services integration gateway (wsig) guide (2005)

3. Chunlin, L., Layuan, L.: An agent-oriented and service-oriented environment for deploying dynamic distributed systems. Computer Standards and Interfaces 24, 323–336 (2002)
4. Dickinson, I.: Nuin: the jena agent framework (2004), http://www.nuin.org
5. Dickinson, I., Wooldridge, M.: Agents are not (just) web services: investigating bdi agents and web services. In: Proc. SOCABE 2005 (2005)
6. Eiter, T., Mascardi, V.: Comparing environments for developing software agents. AI Commun. 15(4), 169–197 (2002)
7. Giorgini, P., Mylopoulos, J., Perini, A., Susi, A.: The tropos metamodel and its use. Informatical journal (2005)
8. Greenwood, D., Lyell, M., Mallya, A., Suguri, H.: The ieee fipa approach to integrating software agents and web services. In: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), Industrial Track (2007)
9. Jack agent platform (2008), http://www.agent-software.com/shared/products/index.html
10. Kostic, M.: Code generation from AML Implementation into CASE tools and support for existing agent platforms. PhD thesis (2006)
11. Marco Mari, M.T., Poggi, A., Turci, P.: Enhancing multi-agent systems with peer-to-peer and service-oriented technologies. In: Sixth International Workshop From Agent Theory to Agent Implementation (AT2AI-6) (2008)
12. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F., Krämer, B.J.: 05462 service-oriented computing: A research roadmap. In: Service Oriented Computing (SOC) (2006)
13. Papazoglou, M.P., van den Heuvel, W.: Business process development lifecycle methodology. Communications of ACM (to appear, 2006)
14. Penserini, L., Perini, A., Susi, A., Mylopoulos, J.: From stakeholder needs to service requirements specifications. Technical report, itc-irst, Automated Reasoning Systems (2006)
15. Singh, M.P., Huhns, M.N.: Service-Oriented Computing Semantics, Processes, Agents. John Wisley and Sons Ltd. (2005)
16. Rafael, M.D., Bordini, H., Winikoff, M.: Current issues in multi-agent systems development (invited paper). In: Post-proceedings of the Seventh Annual International Workshop on Engineering Societies in the Agents World, pp. 38–61 (2007)
17. Rolland, C., Souveyet, C., Kraeim, N.: An intentional view of service-oriented computing. Revue Ingnierie des Systmes dÍnformation (ISI),RSTI (Revue des Sciences et Technologies de lÍnformation)- ISI 13(1), 107–137 (2008)
18. Sudeikat, J., Braubach, L., Pokahr, A., Lamersdorf, W.: Evaluation of agent-oriented software methodologies examination of the gap between modeling and platform (revised selected papers). AOSE-2004 at AAMAS 2004 (2005)
19. Trencansky, I., Cervenka, R.: Agent modelling language (AML): A comprehensive approach to modelling mas. Informatica 29(4), 391–400 (2005)
20. Tsai, W.-T., Wei, X., Paul, R., Chung, J.-Y., Huang, Q., Chen, Y.: Service-oriented system engineering (SOSE) and its applications to embedded system development. In: AOSE 2002 (2007); Revised Papers and Invited Contributions
21. A. UML. Agent uml (2008), http://www.auml.org
22. Witwicki, S.J., Durfee, E.H.: Commitment-based service coordination. In: Kowalczyk, R., Huhns, M., Klusch, M., Maamar, Z., Vo, Q.B. (eds.) Service-Oriented Computing: Agents, Semantics, and Engineering. LNCS, vol. 5006. Springer, Heidelberg (2008)

23. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The gaia methodology. ACM Trans. Softw. Eng. Methodol. 12(3), 317–370 (2003)
24. Zhu, H., Shan, L.: Agent-oriented modelling and specification of web services. In: Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems, vol. 00, pp. 152–159 (2005) (ISBN-ISSN:1530-1443 , 0-7695-2347-1)
25. Zinnikus, I., Benguria, G., Elvester, B., Fischer, K., Vayssire, J.: A model driven approach to agent-based service-oriented architectures. In: Fischer, K., Timm, I.J., André, E., Zhong, N. (eds.) MATES 2005. LNCS (LNAI), vol. 4196, pp. 110–122. Springer, Heidelberg (2006)

A Service-Oriented MultiAgent Architecture for Cognitive Surveillance

David Vallejo, Javier Albusac, Carlos Gonzalez-Morcillo, and Luis Jimenez

Escuela Superior de Informatica,
University of Castilla-La Mancha, Spain
{David.Vallejo,JavierAlonso.Albusac,Carlos.Gonzalez,Luis.Jimenez}@uclm.es
http://oreto.inf-cr.ulcm.es

Abstract. Surveillance systems are being more and more important in a wide variety of environments. In order to obtain better results when analyzing an environment, advanced techniques based on Artificial Intelligence that go beyond segmentation, tracking, and pattern matching are being used. That is, a knowledge layer is needed for improving surveillance. This work describes the architecture of a cognitive surveillance system based on Service-Oriented Principles and Multi-Agent Systems to improve scalability, robustness, and security. Guidelines to expand the surveillance system are covered and the deployment of the architecture in a traffic scenario together with the results obtained are studied.

Keywords: Surveillance, MultiAgent, Service-Oriented.

1 Introduction

Surveillance systems are gradually being introduced in a wide variety of environments. Both the low price and the high performance of hardware and the evolution of the technologies used for carrying out the analysis are contributing to their expansion. Some authors make this evolution explicit by means of different generations of surveillance systems [12]. Currently, the term *third generation surveillance systems* refers to systems designed for dealing with a high number of surveillance resources. Besides, such elements are often geographically distributed over a specific environment. Thus, this type of systems are inherently distributed, not only in the information distribution, but also in the services distribution. Moreover, the notion of distribution goes beyond the physical field, that is, it also covers the semantic one, to create an environment in which different surveillance resources coexist and the information is scattered.

A complex surveillance system should be able to manage a high number of physical devices in a network, with the main goal of providing the user with useful services. In other words, the system should swap the traditional roles assigned to surveillance systems by taking the most active possible role. For example, a security guard may be interested in being warned of a crowd detected by the surveillance system, which may generate a dangerous situation if not detected on time (for example at the end of an escalator). On the other hand, inferring new knowledge may be very interesting. This way, the surveillance system may detect anomalous situations or, particularly, may act accordingly in case of detecting such anomalous situation. These thoughts on surveillance lead us to integrate Artificial Intelligence techniques for surveillance systems being able to behave in an autonomous and active way, increasing the efficiency of such systems. In short, the underlying idea consists of designing high-scalable surveillance systems which adjust to the dynamic nature of surveillance.

To obtain a system which efficiently carries out these functions, it is essential to design it through an approach which assures the interoperability and the adaptation to the different types of services integrated into the environment. However, this approach is not sufficient and it must be supported by an architecture that guarantees the autonomy of such services and the communication needed for the different components which compose the global system to cooperate to obtain good results. Given these premises, we propose the use of Service-Oriented Computing [8] within the context of a Multi-Agent system for dealing with the design of a cognitive surveillance system architecture. The goal pursued is to obtain an architecture that manages the distribution of knowledge from a service-oriented point of view by emphasizing scalability. This way, the architecture allows agents to add services with different features to the surveillance system.

On the other hand, and considering the conceptual perspective of the architecture, it is also very important to take into account several features desired in a surveillance system. For example, the architecture should be robust against failures and secure against attacks and it should facilitate the deployment of services independently of the surveillance environment.

This paper is structured as follows. The following section studies the state of the art and the current research lines in cognitive surveillance systems. Section 3 describes the architecture proposed in this work in depth, studying the different layers of the architecture and analyzing the agents which compose them. Section 4 shows the results obtained in a concrete traffic environment. Finally, Section 5 discusses about concluding remarks and future research lines.

2 Related Work

The union of the service concept and the agent entity as a provider of services involves an interesting approach when dealing with the design of a surveillance system. On the one hand, the service provides the independence and interoperability levels needed for tackling such design from a general point of view, whereas the agent contributes to the autonomy required for the service management. In order to understand the advantages of this union, it is important to study the existing approaches in the state of the art. From an abstract point of view, the system is made up of a number of services. However, the unknown variable of the equation lies in the mechanisms used to support these services and the theoric framework that completes the system architecture.

One of these approaches is centred around the use of a framework which makes easy the deployment of service-oriented systems. The main idea consists of providing tools for creating global perspective service-oriented systems. Within this context, M. Settings presents the SOCAM (*Service-Oriented Context-Aware Middleware*) architecture for developing an architecture based on context-aware services [11]. In this work, the context concept is used to manage the acquisition, the discovery, the interpretation, and the access to the platform services. Within this type of systems, the use of intelligent agents involves a step forward when designing service-oriented systems. Under this approach, L. Chunlin and L. Layuan have developed JASE (*Java-based Agent-oriented and Service-oriented Environment*) with the aim of creating an environment for deploying dynamic distributed systems [4]. Mobile agents are used to support applications, whereas service interface agents are used to wrap services.

The other important approach is related to the term *new generation of grids* [2], also known as *semantic* or *cognitive grids*. This generation is based on a high-level system to create knowledge discovery service-based grids. The architecture of this system is composed of two layers: i) basic services and middleware and ii) services for designing and running knowledge discovery applications. As in the previous approach, there are different works which provide an environment to place such services. M. Cannataro and D. Talia propose the use of a P2P architecture to ensure the grid scalability and decentralize its features to avoid bottlenecks [3]. Another point of view is based on the combination of grids and agents, as considered by Y. Gil [6]. In this work, the author describes the benefits obtained when combining such elements and justifies why grid and agent technologies complement each other.

Until now, the related works perceive the system from a global perspective (making specific instances to meet a particular purpose). On the contrary, this goal can also be derived from the idea of designing a surveillance system with an architecture which facilitates the deployment and the management. Within this context, P. Remagnino et al.'s work stands out [10]. The authors propose a multi-agent architecture to obtain relevant information of scenes from multiple cameras. One of the main ideas is to offer different types of services to a wide variety of users so that they impose restrictions: number, location, and orientation of the cameras, events to identify, and so on. The theoretical framework is based on probability and the authors justify its use due to the uncertain nature of image processing. However, there is no a well-defined architecture which manages special services, such as a service discovery service.

Another important work is developed by B. Abreu et al. [1], in which a layered multi-agent architecture is proposed. The lowest level layer covers sensors and actuators, and a *proxy agent* exists for each camera. Next, and over the previous layer, the object description layer deals with high-level semantic information. In this layer, different agents provide repository, tracking, and classification services. Within this research line, M. Patricio et al. present a more recent work related to third generation surveillance systems [9]. The authors make use of the BDI model to analyze camera images and JADEX to implement the software agents. As in other referred works [10] [1], the authors define *camera agents* for making decisions depending on a simbolic model that represents identified situations and mental states in the form of beliefs, desires, and intentions.

Although the service concept is inherent to surveillance systems, these referred works are not service-oriented and do not provide mechanisms to facilitate the scalability, the discovery, and the composition of services. Faced with this problem, R. Enficiaud et al. propose a generic framework for general purpose surveillance applications against centralized frameworks [5]. The authors suggest an architecture to *store* video surveillance software with the aim of integrating surveillance algorithms through modules which use the framework interfaces.

2.1 Comparison to the Proposed Architecture

The architecture of the surveillance system proposed in this work differs from the related work in three fundamental ideas: i) it is based on surveillance ontologies, ii) it provides explicit mechanisms to add and manage surveillance services, and iii) it combines the multi-agent paradigm with a service-oriented architecture. Firstly, the use of ontologies allows to transparently include surveillance components, that is, the architecture proposed does not change when new surveillance components are needed. This feature is very important because it assures a key concept when designing surveillance architectures: *scalability*. Secondly, the design of specific services to search, manage, and contact with services provided by intelligent agents not only guarantees scalability, but also provides *flexibility*. Finally, the idea of combining multi-agent and service-oriented architectures allows us to deal with the design of a surveillance architecture in a more natural way for two main reasons: i) services captured surveillance demands and made them explicit through *contracts* between service providers and service requestors, and ii) agents involve the *intelligent component* needed for creating cognitive surveillance systems.

3 Architecture

3.1 Architectural Overview

The architecture of the intelligent surveillance system consists of a multi-agent system in which different agents are responsible for managing the services provided by the system from a global perspective, as shown in Figure 1. This section will describe in depth each one of such services and the interactions existing between the agents which provide support to these features.

There are several requirements to assure the efficiency in a service-oriented architecture [8]. Firstly, interoperability between different hardware platforms, operating systems, and programming languages (especially in a surveillance system) should be provided. In this work, ZeroC Ice [7], a modern object-oriented middleware, is used to manage the communication between agents and facilitate the surveillance system deployment. Secondly, a clear, unambiguous, and platform-independent description language should be used for describing the contracts related to the system services. Slice, the interface description language

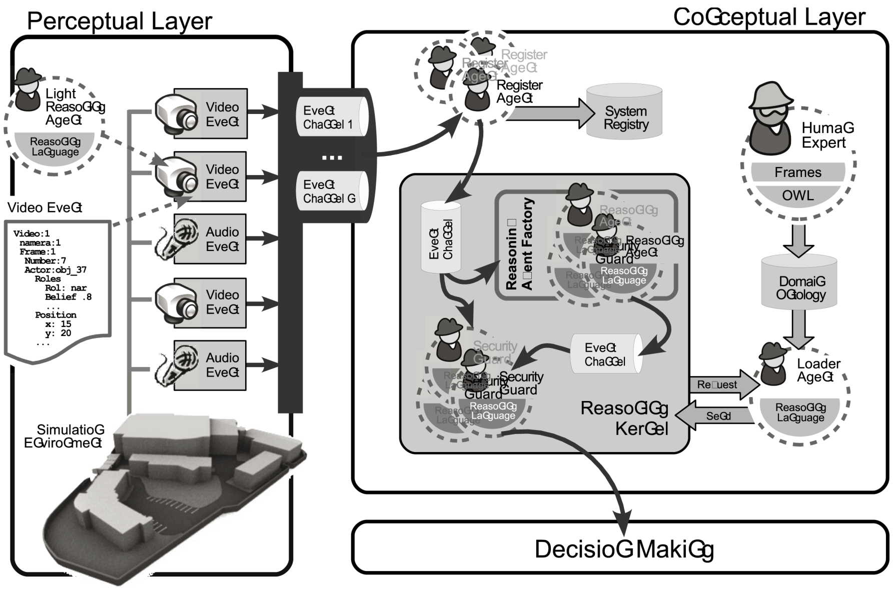


Fig. 1. Architecture of the surveillance system

of Ice, has been chosen for three main reasons: i) independence regards to the type system and the communication protocol, ii) simplicity when defining interfaces, and iii) object-oriented notions. Thirdly, a servide discovery mechanism should be adopted. Fortunatly, the middleware provides us with a native service location mechanism, IceGrid, that can be used to transparently discover services.

Security is another crucial topic when designing surveillance applications to assure privacy, authorization, and authentication. For this reason, the communication carried out by all the components will be done via SSL. In order to manage authorization and authentication, digital certificates will be used to verify the identity of each platform component. In addition, fault tolerance and robustness are covered by transparent replication and load-balancing policies.

The general architecture of the surveillance system is shown in Figure 1. The kernel is represented by the conceptual layer, in which different intelligent agents are responsible for processing the information sent by the perceptual layer. On the other hand, the perceptual layer refers to the information entry point, that is, to the capture of information from the environment. Finally, the decision making layer, of which design goes beyond this work, is fed by the conclusions obtained from the conceptual layer. Both the perceptual and conceptual layers will be covered in depth in the following subsections.

The *Perceptual Layer* is composed of the surveillance equipment which captures or gets information from the surveillance environment. In this context, the main concept is the *event channel*, which refers to a one-way communication channel that allows event publishers and subscribers to communicate. An *event* is considered as a state change in the perception of the scene to be watched. As will be studied in Section 3.2, the architecture has been designed independently of the number of event channels, publishers, and subscribers.

The *Conceptual Layer* shows the real features of the system through the set of services offered to analyze the surveillance environment. As previously described, each one of the services will be held by an agent that provides the communication and cooperation mechanisms needed. The interface of this conceptual layer is represented by the *Register Agent*, which implements the registry service to make persistent the information obtained from the perceptual layer. The design of this layer is based on scalability and, in particular, on the *Reasoning Agent Factory*. Through this component, the system enables to add *Reasoning Agents* when needed. Both components and the *Security Guard* compose the *Reasoning Kernel*. This last agent (the Security Guard) reasons from a global perspective, that is, from the conclusions obtained by the different reasoning agents. In other words, the main service provided by this agent consists of reasoning on the global normality of the scene. The system design also allows different reasoning agents to use different knowledge representation languages thanks two main reasons: i) the use of event channels and ii) the use of a neutral description language.

3.2 Perceptual Layer

The perceptual layer is composed of all the surveillance equipment which gets information from the environment and notifies it to the system. Such devices are classified into video capture, e.g. a camera, audio capture, e.g. a microphone, and sensors, such as a scanner. In this architecture the abstraction of event generator will be used to refer to any surveillance device. In this context, the event channel is considered as a one-way communication channel that allows event generators to send information to those elements interesting in processing it. Therefore, two roles are identified: the publisher and the subscriber. The system architecture has been designed independently of the number of publishers, subscribers, and event channels to assure scalability thanks to these last channels, as they provide the logical separation between publishers and subscribers.

As introduced in Section 3.1, an event is defined as a state change of the surveillance environment. Events are assumed to be independent and they are provided by vigilance devices. Examples of events are a sound captured by a microphone, a car on a parking, or the information sent by a scanner used by a person. In order to represent and send events homogeneously, a language that makes easy descriptions should be adopted. Slice, the interface description language of Ice, has been chosen for two main reasons: *simplicity* and *efficiency*. Simplicity refers to use a fast, clear, and simple mechanism for defining interfaces and data types. Efficiency refers to use the inherent language of the middleware to avoid conversions to other languages. For example, the information of a video event is structured in different elements:

– *id*: the object identifier.
– *roles*: the possible roles of the object, that is, a sequence of class-belief pairs. For example, a certain object can be classified into a car with a belief of 0.8 and into a motorbike with a belief of 0.2.

– *pos*: the 2D position of an object in a concrete moment retrieved from a camera frame.
– *t*: the moment in which the object information was retrieved.
– *r1-r2*: the object size defined by the radiuses of the ellipsis in which the object is allocated.
– *speed*: the object vector speed measured in pixels per second.

3.3 Conceptual Layer

The conceptual layer represents the kernel of the architecture because it is responsible for processing information to obtain knowledge, that is, to transform the surveillance system from a passive component to an active one. Under this approach, the system will be able to activate alarms when detecting anomalous situations. There are different agents which provide and request services when needed. Besides, the communication model remains consistent thanks to the use of event channels to sent information. The basic workflow of the surveillance system (see Figure 2) consists of several steps: i) a new event is received from the perceptual layer; ii) the Register Agent stores the event content in the System Registry; iii) the Reasoning Agents subscribed to the event channel in which the event was published process the information, obtain new knowledge, and activate possible alarms; iv) the Security Guard studies the conclusions generated by the reasoning agents. Next, the interfaces, services, and interactions of each agent will be covered in depth.

The *Register Agent* represents the entry point of the conceptual layer and acts as a subscriber for all the events published by the devices of the perceptual layer. The interface of the Register Agent is defined by the *notifyEvent* operation, which is used to receive events. Currently, we are dealing with video events because they provide us more information than other types of events. The main service offered by this agent consists of making persistent all the events. In other words, the Register Agent provides a storage service through the System Registry. This component represents the database of the surveillance system which will be used to add a query service to the Register Agent.

The *Reasoning Kernel* is composed of two types of reasoning agents, the *Reasoning Agent* and the *Security Guard*, and one component responsible for the management of the reasoning agents. This last component, named *Reasoning Agent Factory*, implements the abstract factory pattern and provides operations for creating, searching, and listing reasoning agents. When a reasoning agent is instantiated through the Reasoning Agent Factory, it automatically obtains references to the event channels known by the Reasoning Agent Factory. This way, the reasoning agents are able to process events as soon as they are born. The factory has also been designed after carefully thinking about concurrency by interlocking the *create*, *find*, and *list* operations. In relation to the deployment, this service has been implementing through the IceBox service of the middleware ZeroC Ice, which facilitates its administration and configuration. The reasoning agents created by the Reasoning Agent Factory are distinguished for the identifiers specified when invoking the *create* operation.

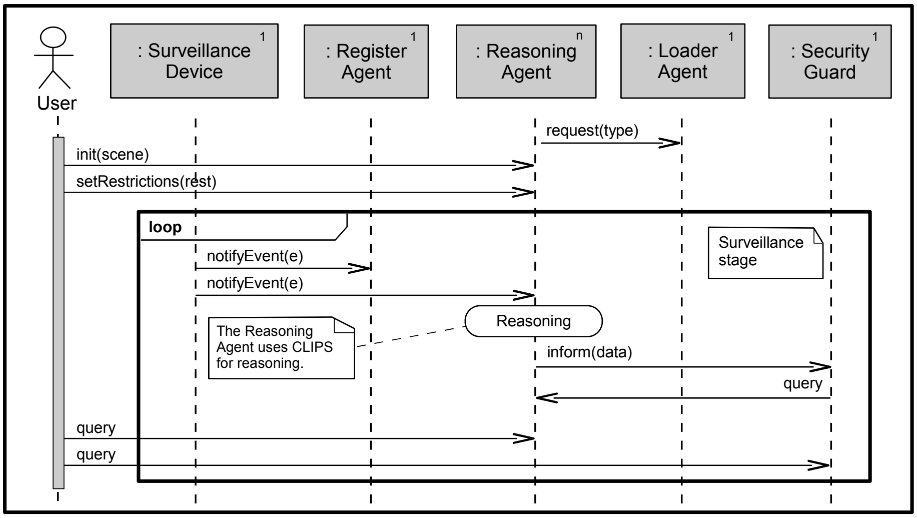


Fig. 2. Interactions between the system components

The *Reasoning Agent* has been designed for reasoning on a certain normality concept. This way, different reasoning agents which reason on different concepts or topics can coexist in a concrete moment. The entity responsible for integrating all the knowledge obtained from these agents is represented by the *Security Guard*. The reasoning mechanism is a two-level mechanism in which the Security Guard receives information from the reasoning agents and these agents process the events generated by the perceptual layer. For example, a reasoning agent may deal with the normality about object movements. Thus, this agent may detect an anomalous behaviour if an object has not associated movements, that is, if such object is making an unrecognized movement (e.g. a car driving along a pedestrian street). When a reasoning agent is instantiated, it requests the ontologies needed for reasoning on a concept to the *Loader Agent*. Depending on such concept, this agent will send one or another domain ontology. This Loader Agent also represents the interface between the human expert and the surveillance system. Currently, we are using CLIPS both for representing the knowledge and for reasoning on the defined knowledge.

We are dealing with knowledge at two levels: environment level and normality concept level. On the one hand, the environment level refers to general knowledge about surveillance. This knowledge covers concepts such as positions, zones, intervals, or actors. On the other hand, the normality concept level refers to knowledge about normality concepts. In order to assure scalability, each agent will reason on a normality concept by using the general knowledge about surveillance and the particular knowledge about the normality concept in which it is specialized. For example, a reasoning agent specialized in movements may deal with spatial restrictions, associated movements, recognized movements, and so on.

As previously mentioned, a reasoning agent is specialized in a normality concept. It is necessary to specify or notify the information related to the surveillance environment before starting the analysis. First, the general information is sent to the reasoning agent, that is, the scene description. A scene is divided into three components: i) the set of zones which composes the scene, ii) the movements which can be carried out by the objects in such scene, iii) and the time intervals related to the scene (e.g. the working day). The next step consists of defining the restrictions associated to the scene. These restrictions are spatial restrictions, temporal restrictions, and actor restrictions. Such general restrictions are specialized for each normality concept, that is, there are spatial, temporal, and actor restrictions for the normality about movements, for instance, which are (or not) different from other concept to be analized (the normality about velocity, for example). Finally, and knowing the scene description and the restrictions about a normality concept, a reasoning agent is able to analyze the normal behaviours about a certain concept. The Reasoning Agent also implements a query service which can be used to know if the normality of a concept is being carried out by the objects and with which belief. This query information is similarly related to the normality concept linked to the reasoning agent.

Finally, and at a higher abstraction level, the *Security Guard* will process all the information sent by the other reasoning agents. This information is received through an internal event channel. The main idea will consist of analyzing more complex behaviours which take into account various normality concepts. Besides reasoning at a higher abstraction level, this agent will communicate with the decision making layer to act against anomalous behaviours.

Until now, fault tolerance and robustness are topics that have not been covered. Both concepts are essential in a surveillance systems because if the system goes down, then surveillance disappears. To overcome this potencial problem, we have adopted a mechanism based on transparent replication which also supports load balancing policies. In other words, different instances of the same agent (Register Agent, Reasoning Agent, or Security Guard) may coexist in different computers (or in the same). Moreover, there are various load balancing types that can be used by defining a configuration parameter: random, adaptive, round robin, and order. We are currently using the adaptive one, which transparently chooses the least-loaded agent in terms of system load. Security, authorization, and authentication are covered by SSL and a PAM-based permissions verifier.

3.4 Scaling the Surveillance System

This subsection resumes the design solutions applied in the proposed architecture when scaling the surveillance system, in particular by taking into account three topics: i) surveillance infrastructure, ii) new normality concepts for reasoning, and iii) applications demands.

New hardware is often added to a surveillance system when its deployment has been done. For example, a new camera may be added to a critical point of security when the system is running. In the proposed architecture, the solution simply consists of making that the new hardware acts as a event publisher in one or more desired event channels. This way, the subscribers will be notified with the information captured by the new surveillance equipment.

The second important question lies in adding a new reasoning agent. In this case, various tasks must be done: to specify the data types needed for communicating the information of the new normality concept (restrictions and queries), to define the knowledge related to the new concept (e.g. about the working hours of different workers) and load it into the Loader Agent, and to instantiate the new reasoning agent through the Reasoning Agent Factory.

Finally, the increase of the number of surveillance devices and normality concepts related to the reasoning layer may require more computation, that is, more powerful servers or a higher number of servers. In the last case, the architecture solves this constraint thanks to the transparent replication provided by the middleware. In both cases, the proposed solution also consists of using the service of the middleware that provides the distribution of server executables and dependent files.

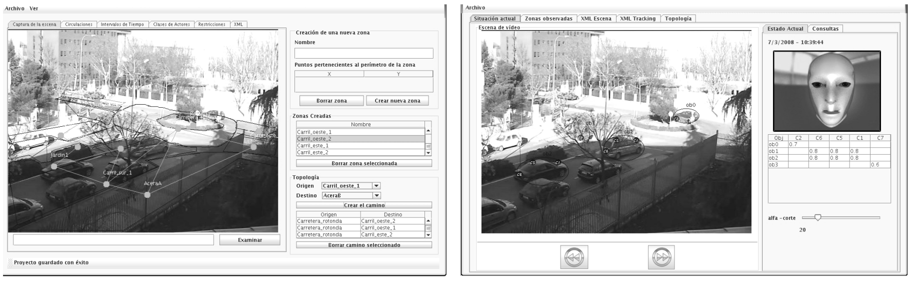
4 Deployment and Results

This section describes the deployment of the proposed surveillance architecture in the traffic scenario shown in Figure 4. Within this context, we have carried out the analysis depending on the normality about vehicle and pedestrian movements. In other words, the main goal of this deployment consists of studying if both pedestrians and vehicles behave correctly when doing their movements. With this purpose, we have developed a Movement Reasoning Agent and defined a movement ontology in CLIPS. This agent's behaviour is summarized in Algorithm 1.

Most of the concepts of the ontology are heavily based on uncertainty. This is due to the dependence on the perceptual layer. Currently, recognition and identification techniques cannot assure a precise classification due to internal (algorithms) and external (e.g. adverse weather conditions) questions. For these reasons, we associate beliefs to concepts such as the own objects, the zones covered, or the fulfilment of a spatial restriction. These topics are seen in Algorithm 1 when a global belief is related to an associated movement and calculated from the belief of its restrictions (see line 14 of Algorithm 1).

In order to test the proposed architecture and debug the results obtained in a concrete environment, we have developed two tools that make easy to define knowledge related to a certain domain and to debug the surveillance results. After briefly describing such tools, the experimental results obtained in the traffic environment and the evaluation of such results will be studied.

Figure 3 (left) shows the interface of a knowledge acquisicion tool to help the human expert to define the knowledge domain. This tool allows to specify zones, movements, and restrictions over these movements. Figure 3 (right) also shows the interface of a second tool in which the video stream and the tracking of the mobile objects of the environment are monitored. In fact, the tool allows us to check if the reasoning is correct. This way, the user can make use of two buttons



Proyecto guardado con éxito

X	Y

Nombre
Carril_oeste_1
Carril_oeste_2
Carril_este_1
Carril_este_2

Origen: Carril_oeste_1
Destino: AceraB

Origen	Destino
Carretera_rotonda	Carril_oeste_2
Carretera_rotonda	Carril_oeste_1
Carretera_rotonda	Carril_este_2

7/3/2008 - 10:39:44

Obj	C2	C6	C5	C1	C7
ob0	0.7				
ob1		0.8	0.8	0.8	
ob2		0.8	0.8	0.8	
ob3					0.6

alfa - corte: 20

Fig. 3. Left: Knowledge acquisition tool. **Right:** Debugging tool.

Algorithm 1. Movement Reasoning Agent's Behaviour

```
 1: objects ⇐ objects whose position has changed
 2: for obj in objects do
 3:    current-zones ⇐ get-zones-object(obj)
 4:    obj.update(current-zones)
 5:    for mov in defined-movements do
 6:       if mov.beginning in current-zones and mov not associated then
 7:          obj.associated-movements ⇐ mov
 8:       end if
 9:    end for
10:    for mov in actor.associated-movements do
11:       mov.belief-act ⇐ check-actor-restrictions(mov)
12:       mov.belief-spa ⇐ check-spatial-restrictions(mov)
13:       mov.belief-tmp ⇐ check-temporal-restrictions(mov)
14:       mov.belief ⇐ update-belief(mov.belief-act, mov.belief-spa, mov.belief-tmp)
15:       if mov.finish() then
16:          obj.recognized-movements ⇐ mov
17:          obj.associated-movements.delete(mov)
18:       end if
19:    end for
20: end for
```

to move over time. The left arrow button permits the user to study past frames while the righ arrow button refers to the next frames. Besides, the interface shows tables which resume the information of the scene objects, the associated and recognized movements, and the beliefs of each movement. Such beliefs are obtained in relation to the fulfilment of the restrictions. This tool also allows the user to explicitly query the reasoning kernel.

In order to test the architecture, we have chosen a traffic scenario familiar with us. This scenario refers to the exterior of the Superior School of Computer Science at the University of Castilla-La Mancha (Spain) and the images have been captured from the second floor. As shown in Figure 4, the scenario is mainly composed of two pavements in which pedestrian walk, landscaped areas

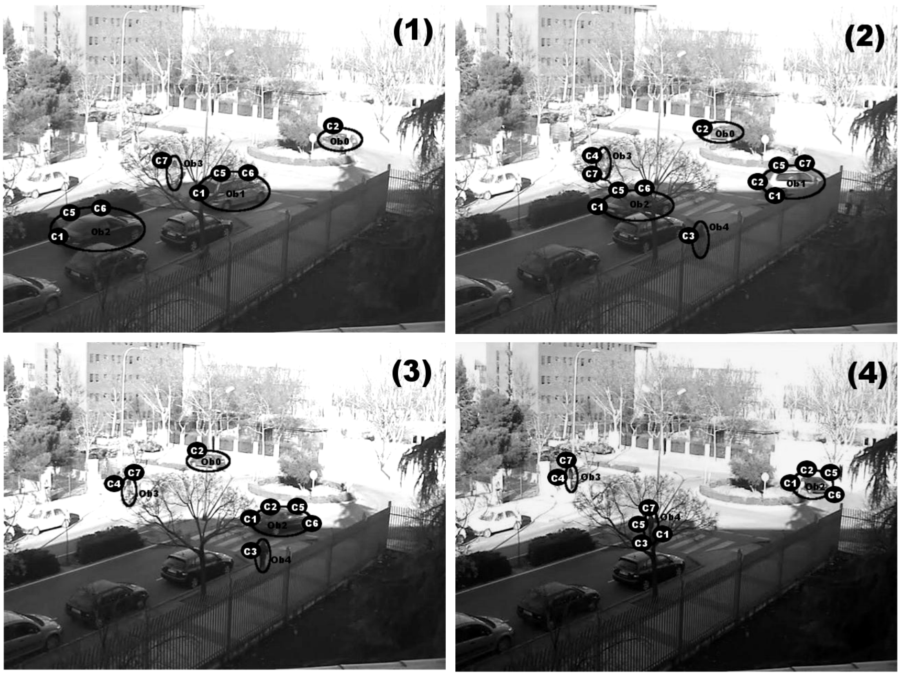
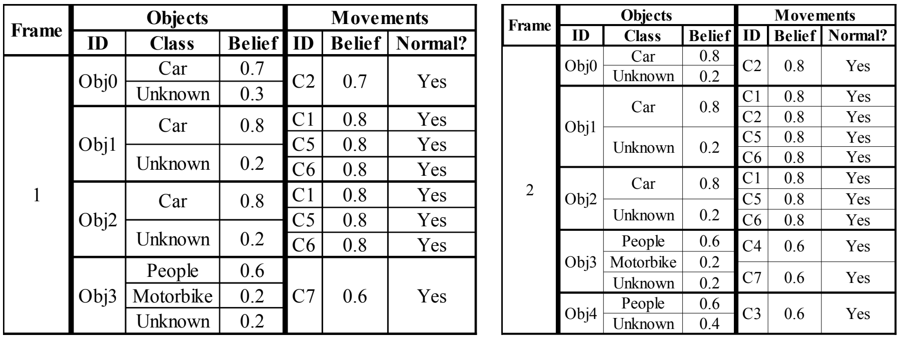


Fig. 4. Key frames of the traffic scenario

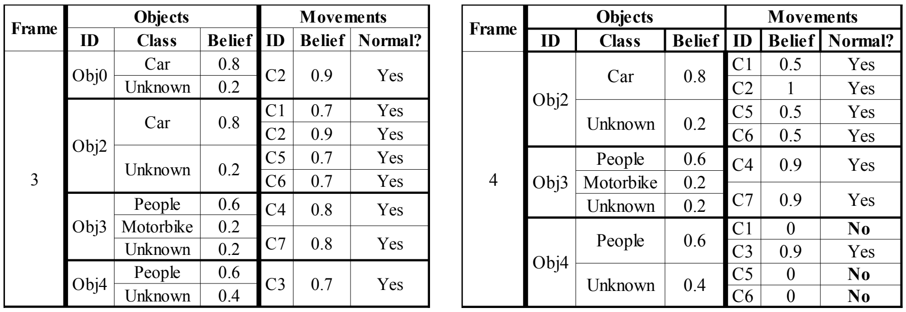


Frame	Objects			Movements		
	ID	Class	Belief	ID	Belief	Normal?
1	Obj0	Car	0.7	C2	0.7	Yes
		Unknown	0.3			
	Obj1	Car	0.8	C1	0.8	Yes
				C5	0.8	Yes
		Unknown	0.2	C6	0.8	Yes
	Obj2	Car	0.8	C1	0.8	Yes
				C5	0.8	Yes
		Unknown	0.2	C6	0.8	Yes
	Obj3	People	0.6	C7	0.6	Yes
		Motorbike	0.2			
		Unknown	0.2			

Frame	Objects			Movements		
	ID	Class	Belief	ID	Belief	Normal?
2	Obj0	Car	0.8	C2	0.8	Yes
		Unknown	0.2			
	Obj1	Car	0.8	C1	0.8	Yes
				C2	0.8	Yes
		Unknown	0.2	C5	0.8	Yes
				C6	0.8	Yes
	Obj2	Car	0.8	C1	0.8	Yes
				C5	0.8	Yes
		Unknown	0.2	C6	0.8	Yes
	Obj3	People	0.6	C4	0.6	Yes
		Motorbike	0.2	C7	0.6	Yes
		Unknown	0.2			
	Obj4	People	0.6	C3	0.6	Yes
		Unknown	0.4			

Fig. 5. Frames 1 and 2 of the traffic surveillance test

which separate the two one-way lanes in which vehicles drive, and a roundabout that both vehicles and pedestrian cannot invade. The definitions of vehicle normal movements exclude vehicles driving in the opposite direction or invading landscaped areas. In the case of pedestrian, the normal movements are related to walk along the pavement or to cross over the pedestrian crossing. Figures 5 and 6 resumes the scene configuration in each one of the key frames shown in Figure 4. These tables expose object and movement information in each frame.



Frame	Objects			Movements		
	ID	Class	Belief	ID	Belief	Normal?
3	Obj0	Car	0.8	C2	0.9	Yes
		Unknown	0.2			
	Obj2	Car	0.8	C1	0.7	Yes
				C2	0.9	Yes
		Unknown	0.2	C5	0.7	Yes
				C6	0.7	Yes
	Obj3	People	0.6	C4	0.8	Yes
		Motorbike	0.2	C7	0.8	Yes
		Unknown	0.2			
	Obj4	People	0.6	C3	0.7	Yes
		Unknown	0.4			

Frame	Objects			Movements		
	ID	Class	Belief	ID	Belief	Normal?
4	Obj2	Car	0.8	C1	0.5	Yes
				C2	1	Yes
		Unknown	0.2	C5	0.5	Yes
				C6	0.5	Yes
	Obj3	People	0.6	C4	0.9	Yes
		Motorbike	0.2	C7	0.9	Yes
		Unknown	0.2			
	Obj4	People	0.6	C1	0	**No**
				C3	0.9	Yes
		Unknown	0.4	C5	0	**No**
				C6	0	**No**

Fig. 6. Frames 3 and 4 of the traffic surveillance test

Object information refers to the object ID and the set of classes associated to the object. The belief represents the certainty of the object belonging to a class. Movements also have a belief that is calculated from the set of restrictions linked to a concrete movement. If the belief of a movement does not overcome a predefined threshold, then the movement is not considered normal. Higher values of this threshold refers to a stricter surveillance.

Table 1. 60-second tests related to the scenario of Figure 4

Test	Situations	Normal sit.	Anomalous sit.	Errors	Efficiency
1	24	24	0	0	100%
2	72	72	0	0	100%
3	154	129	0	25	0.83%
4	61	61	0	5	0.91%
5	80	73	7	0	100%

Table 1 shows 5 tests of the traffic environment. Some of them are related to uncommon situations. For example, test number 5 simulates a car going onto the roundabout. As this movement has not been defined, the system infers that the situation is not normal. Tests 3 and 4 generate errors because there are some people who walk along the nortern pavement and this movement was not previously defined. Therefore, the system identifies anomalous situations which should be normal. Test 3 generates more situations due to a high number of people. As these people move slow, the number of potential movements is greater than in the other tests.

5 Discussion and Conclusion

The evolution of surveillance systems is demanding for architectures that assure scalability, flexibility, and robustness. Besides, such evolution tends to be knowledge-oriented, that is, more recent surveillance systems are making use of Artificial Intelligence methods to improve results or to add new ones. The proposed architecture in this paper offers a new perspective when designing cognitive surveillance systems due to the use of Service-Oriented Principles and Multi-Agent Systems. In addition, key features of surveillance, such as scalability, fault tolerance, robustness, and security are covered by making use of the middleware services.

Experimental results are based on analyzing the movements done by the different objects that appear in a traffic scene. These results prove that the conceptual layer allows to obtain new information when analyzing the environment, that is, if a concrete object is doing a correct movement with a certain belief. On the other hand, the architecture design offers several desirable features in a surveillance system:

– On-demand service discovery thanks to the use of a explicit service manager.
– Scalability to other reasoning concepts thanks to the Reasoning Agent Factory.
– Transparency related to the number of surveillance devices due to the use of event channels.
– Fault tolerance, robustness, authorization, authentication, and security as a consequence of using advanced middleware services such as transparent replication, load balancing, digital certificates, and SSL.

The use of human expert knowledge in a explicit layer of the surveillance architecture opens numerous research lines. Our current work is focused on adding more normality concepts to reason in real time and expanding the system to other domains. Within this context, different reasoning agents will coexist in the proposed architecture. We are also planning to optimize some critical parts related to geometric calculus by using graphic processing units. On the other hand, another important research line is related to migrate a reasoning agent from the server to the camera itself. This way, the network latency will be removed and the *light reasoning agents* instantiated in the cameras will make the surveillance more efficient. Another relevant question is reusability, that is, adapting the architecture to other domains than surveillance. In this context, we are considering certain models such as those proposed by the FIPA Nomadic Agent Working Group.

Acknowledgments

This work has been funded by the Regional Government of Castilla-La Mancha under the Research Projects PBC-06-0064 and PAC-06-0141 and by the Spanish Ministry of Education and Science (TIN2007-62568).

References

1. Abreu, B., Botelho, L., Cavallaro, A., et al.: Video-based multi-agent traffic surveillance system. In: IEEE Intelligent Vehicles Symposium, pp. 457–462 (2000)
2. Cannataro, M., Talia, D.: Knowledge grid: An architecture for distributed knowledge discovery. Communications of the ACM 46(1), 89–93 (2003)

3. Cannataro, M., Talia, D.: Semantics and knowledge grids: building the next-generation grid. IEEE Intelligent Systems 19(1), 56–63 (2004)
4. Chunlin, L., Layuan, L.: An agent-oriented and service-oriented environment for deploying dynamic distributed systems. Computer Standards & Interfaces 24(4), 323–336 (2002)
5. Enficiaud, R., Lienard, B., Allezard, N., Sebbe, R., Beucher, S., Desurmont, X., Sayd, P., Delaigl, J.: Clovis-a generic framework for general purpose visual surveillance applications. In: IEEE Workshop on Visual Surveillance, pp. 177–184 (2006)
6. Gil, Y.: On agents and grids: Creating the fabric for a new generation of distributed intelligent systems. Web Semantics: Science, Services and Agents on the World Wide Web 4(2), 116–123 (2006)
7. Henning, M.: A new approach to object-oriented middleware. Internet Computing, IEEE 8(1), 66–75 (2004)
8. Huhns, M.N., Singh, M.P.: Service-oriented computing: key concepts and principles. IEEE Internet Computing 9(1), 75–81 (2005)
9. Patricio, M.A., Carb, J., Prez, O., Garca, J., Molina, J.M.: Multi-Agent Framework in Visual Sensor Networks. EURASIP Journal on Advances in Signal Processing, 1–21 (2007)
10. Remagnino, P., Shihab, A.I., Jones, G.A.: Distributed intelligence for multi-camera visual surveillance. Pattern Recognition 37(4), 675–689 (2004)
11. Settings, M.: A service-oriented middleware for building context-aware services. Journal of Network and Computer Applications 28(1), 1–18 (2005)
12. Valera, M., Velastin, S.A.: Intelligent distributed surveillance systems: a review. IEE Proceedings Vision, Image and Signal Processing 152(2), 192–204 (2005)
13. Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and Practice. Knowledge Engineering Review 10(2), 115–152 (1995)

Trust-Based Classifier Combination for Network Anomaly Detection

Martin Rehák[1], Michal Pěchouček[1], Martin Grill[1,2], and Karel Bartos[1,2]

[1] Department of Cybernetics and Center for Applied Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague
Technická 2, 166 27 Prague, Czech Republic
{mrehak,pechouc}@labe.felk.cvut.cz
[2] CESNET, z. s. p. o.
Zikova 4, 160 00 Prague, Czech Republic
{bartosk,grillm}@labe.felk.cvut.cz

Abstract. We present a method that improves the results of network intrusion detection by integrating several anomaly detection algorithms through trust and reputation models. Our algorithm is based on existing network behavior analysis approaches that are embodied into several detection agents. We divide the processing into three distinct phases: anomaly detection, trust model update and collective trusting decision. Each of these phases contributes to the reduction of classification error rate, by the aggregation of anomaly values provided by individual algorithms, individual update of each agent's trust model based on distinct traffic representation features (derived from its anomaly detection model), and re-aggregation of the trustfulness data provided by individual agents. The result is a trustfulness score for each network flow, which can be used to guide the manual inspection, thus significantly reducing the amount of traffic to analyze. To evaluate the effectiveness of the method, we present a set of experiments performed on real network data.

1 Introduction

This paper presents a specific application of techniques from agent trust modeling in the domain of Network Intrusion Detection and shows how to apply these techniques to combine several intrusion detection methods and improve the quality of their decisions.

The purpose of the *Network Behavior Analysis* (NBA) systems [1] is to identify the attacks against the network infrastructure and hosts by observing the significant events in the structure and volume of network traffic. Most of the NBA systems are based on *anomaly detection* [2] principles – they build the model of the traffic in the network from the past observations, predict the properties of current traffic and identify the potentially malicious actions by comparing the prediction with the observed traffic.

The proposed method is based on network observation using the NetFlow or IPFIX data that may be provided by routers (the NetFlow protocol was originally defined by Cisco [3]) or specialized devices [4]. Each **flow** corresponds to one direction of TCP connection (or UDP/ICMP equivalent); all observed packets with the same source IP address (*srcIP*), source port (*srcPrt*), destination address (*dstIP*), destination port (*dstPrt*) and protocol (TCP/UDP/ICMP) constitute the flow. In addition to these definition parameters, we can observe supplementary data, such as number of bytes and packets, duration of the flow and other parameters. The NetFlow data is aggregated over an observation period, typically a 5 minute interval. Once aggregated, the NBA system extracts relevant features of the data and concludes which flows are malicious, and which are part of the legitimate network traffic.

The usefulness of current NBA systems is severely impacted by two major shortcomings: their limited effectiveness, i.e. high error rate [5,6], and relatively low efficiency, which does not allow their deployment on high bandwidth network links. The *effectiveness* of a particular IDS system is described by two values, the ratios of false positives and false negatives. The **false positives** are the legitimate, non-malicious flows that are classified as malicious, while the **false negatives** are the malicious flows classified by the system as legitimate. We will describe the *efficiency* of the IDS system in terms of the number of network flows per second it can process, as this value is directly linked with network bandwidth.

Our work uses the techniques developed in the field of agent-based trust and reputation modeling to improve the effectiveness of the system. Furthermore, the deployment within an efficient agent platform supports natural parallelization of tasks and thus easy distribution across multiple processor cores. Specifically, we improve the error rate of the collective detection system by:

- separation of short term anomaly detection and long-term trust modeling
- collaboration between heterogeneous trusting agents, with following processes performed on the top of the original anomaly detection methods:
  - anomalies are considered by individuals only if they are consistently identified by anomaly detection models of majority of agents
  - anomalies are considered only if the flows that constitute them fall into one or few traffic classes in the trust models of individual agents
  - reputation mechanism integrates the trustfulness data from several agents

In Section 2, we will discuss the necessary extension of trust modeling techniques, before presenting the anomaly detection algorithms in Section 3.1 and the core contribution of this work in the remainder of the Section 3. We evaluate our solution on real data in the experiments described in Section 4, and discuss the related work before concluding.

2 Extended Trust Modeling

Trust models [7,8,9,10] are specialized knowledge structures designed to maintain information about the trustworthiness of the partners, either acquired from agent's own interactions, observed from the interactions of others, or received from other agents by means of reputation mechanism [11]. The design of trust models emphasizes features such as fast learning, robustness in response to false reputation information [12] and robustness with respect to environmental noise. Extended trust models [13,14] are used to address several important assumptions of trust models and to make them more relevant for practical deployment, as they are able to:

- include the **context** of the trusting situation into the reasoning, making the trust model situational,
- use the similarities between trustees and situations to infer their trustfulness during the **first encounter**, and
- protect the model against trustee **identity changes**.

Extended trust models are inspired by machine learning [14] and pattern recognition [13] approaches. The models achieve the above-listed goals by reasoning not about the performance of specific agents, but about the trustfulness of more general identities situated in a specific context, which describes the trusting situation. More specifically, each trustee and trusting situation are described by a set of relevant observable features (feature vector). The features included in the feature vector define the feature space, a metric space on which the trust model of each agent operates. Trustfulness is determined for significant clusters in this space (i.e. reflects the behavior of a class of similar agents in a similar situation), and Section 3 describes the update and query operations in more detail.

In the network security domain, low *trustfulness* of the flow means that the flow is assumed to be malicious, i.e. a part of an attack. Trustfulness is determined in the $[0, 1]$ interval, where 0 corresponds to complete distrust and 1 to complete trust. The *identity* of each flow is defined by the features we can observe directly on the flow: *srcIP, dstIP, srcPrt, dstPrt, protocol*, number of *bytes* and *packets*. If two flows in a data set share the same values of these parameters, they are assumed to be identical. The *context* of each flow is defined by the features that are observed on the other flows in the same data set, such as the number of similar flows from the same *srcIP* [15], or entropy of the *dstPrt* of all requests from the same host as the evaluated flow [16]. Identity and context features are used to define the feature space for each specific type of agent introduced below.

3 Detection Process

The detection functionality is distributed among the **detection agents** that cooperate to improve their classification performance. Each agent is based on a specific anomaly detection method (detailed below), and each agent's method also defines the features used to represent the context of the flow in its trust model. Therefore, the models differ between the agents, and we can not perform direct translation between them. Specificity of the trust model and anomaly detection method improves anomaly detection results, reduces the dimensionality of the problem and reduces the computational requirements of the trust model (for each agent). It also directly contributes to elimination of false positives, as we will discuss in Section 3.3. On the downside, model specificity limits the collaboration to those stages in the process when the agents can use a common language to share the anomalies or reputation values relative to individual flows.

In the presented system configuration, all agents process the same network data. At the end of each observation interval j, all detection agents $X \in Ags$ receive the same input set Φ_j of network flows $\varphi_{i,j}$, and this data is the only algorithm input for each observation period j. The processing is performed in three stages (Fig. 1):

– anomaly detection,
– trust model update, and
– flow classification by individual and collective trustfulness determination

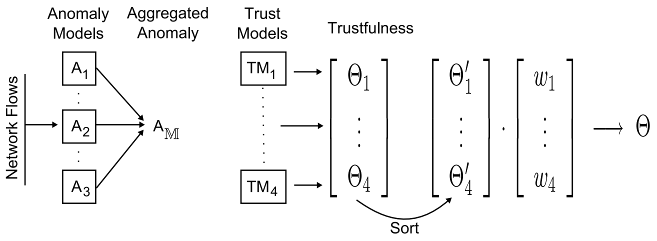


Fig. 1. Detection process overview

Before the detailed discussion of these stages, we will introduce the most important terms:

Anomaly $A_X(\varphi_{i,j})$ is a number in the $[0, 1]$ interval describing agent's X opinion about the anomaly of the flow $\varphi_{i,j}$ in the current set Φ_j. One represents the maximal anomaly, and zero no anomaly at all. It is provided by the anomaly detection method embedded in the detection agent X.

Trustfulness $\Theta_X(\varphi_{i,j})$ value can be determined for any feature vector in the feature space of agent X. It falls into the $[0, 1]$ interval as well, and it indicates the estimated level of maliciousness. Flows with trustfulness close to 0 are considered to be malicious, while the flows with high trustfulness are classified as legitimate (trusted).

Feature vector $ix_X(\varphi_{i,j})$ represents the identity and context features determined for the flow $\varphi_{i,j}$ by agent X in the feature space. We use the term *centroid* to denote the permanent feature vectors r_k that are positioned in the feature spaces of trusting agents. The centroids act as trustees of the model, and the trustfulness value $\Theta(r_k)$ of each centroid is updated with relevant observations, and used to deduce the trustfulness of feature vectors in its vicinity.

Metrics $dist_X(ix_X(\varphi_{i,j}), ix_X(\varphi_{k,l}))$ determines the distance of two feature vectors in the feature space of agent X. As mentioned above, each agent type has a metrics defined by its context representation (and the shared identity part), and we shall emphasize that: $dist_X(ix_X(\varphi_{i,j}), ix_X(\varphi_{k,l})) = dist_Y(ix_Y(\varphi_{i,j}), ix_Y(\varphi_{k,l}))$ almost never holds for $X \neq Y$.

3.1 Detection Agent Types

This section briefly presents the anomaly detection techniques and traffic features used by most important types of detection agents in the system. All agents use the same representation of flow identity, but differ in the context dimensions of the feature space, as we will describe below. The feature space distance function (metrics) is a sum of two components, one covering the identity subspace, the other context subspace dimensions. The identity component is identical for all agent types, and type-dependent context distance is described with each agent type below:

MINDS algorithm [15] builds the context information for each flow using the: number of flows from the same source as the evaluated flow, number of flows towards the same destination host, number of flows towards the same destination from the same source port, and number of flows from the same source towards the same destination port. This makes the context space four dimensional, with logarithmic distance scale in each dimension, combined into the global distance as a sum of their squares. Contrary to the original work, we judge the anomaly from the difference between the floating average of past values and observation in each of the four context dimensions.

Xu *et al.* [16] actually classifies the traffic sources, which imposes the same context for all flows from the same *srcIP*. For each source, we determine the normalized entropy of the set of source ports, destination ports and destination IPs of all the flows from this source, thus defining a 3D context. Anomalies are then determined by application of static classification rules that divide the traffic into normal and anomalous classes. Distance between the contexts of two flows is computed as a difference between the 3 normalized entropies of each flow, combined as sum of squares.

Volume prediction algorithm [17] uses the Principal Components Analysis to build the model of traffic volumes from individual sources in number of bytes, packets and flows. Then, it identifies the difference between the predicted and real traffic for each source IP and all flows from the source are assigned this value transformed into the $[0, 1]$ interval as anomaly. Again, the context is identical for all flows from the same source, and is defined by the difference between the predicted and real number of flows, packets and bytes from the *srcIP* of the flow. Distance in each of the 3 context dimensions is logarithmic, combined as a sum of squares.

Entropy prediction algorithm [18] works in exactly the same manner as the previous type, but predicts the entropies of *dstIP*, *dstPrt* and *srcPrt* instead of traffic volumes. To aggregate the distance in the context subspace, we determine the distance between the residual entropies as an absolute value of their difference, and add their squares. For detailed discussion of agent types and modifications of original algorithms, please refer to our previous publication [19].

3.2 Collective Trust Modeling

After the introduction of terms and brief description of anomaly detection and traffic representation techniques, we will describe the stages of the algorithm:

Anomaly detection. During the anomaly detection stage, each individual agent A uses its embedded anomaly detection method to determine the anomaly $A_A(\varphi_{i,j})$ of each flow of the set Φ_j. As the features (dimensions) of the feature space of agent A are identical to those used by its anomaly detection method, we can (informally) write $A_A(\varphi_{i,j}) = A_A(ix_A(\varphi_{i,j}))$ to emphasize that the information in the feature vector $ix_A(\varphi_{i,j})$ of the flow is sufficient to determine its anomaly. The anomaly values are shared with other detection agents, and used as an input in the second phase of the processing – all agents thus have the same aggregated anomaly value $A_{Ags}(\varphi_{i,j})$ for each flow, shown in Eq. 1, and this value averages the anomaly opinions of all detection agents:

$$A_{Ags}(\varphi_{i,j}) = \frac{1}{|Ags|} \sum_{X \in Ags} A_X(ix_X(\varphi_{i,j})) \tag{1}$$

Trust update. During the trust update, the agents integrate the anomaly values of individual flows from the set Φ_j into their trust models. As the reasoning about the trustfulness of each individual flow is computationally infeasible and unpractical (the flows are single shot events by definition), the extended model holds the trustfulness $\Theta(r_k)$ of centroids r_k (significant flow samples, e.g. centroids of fuzzy clusters) in the feature space, and the anomaly $A_{Ags}(\varphi_{i,j})$ of each flow $\varphi_{i,j}$ is used to update the trustfulness of centroids in its vicinity. Eq. 2 specifies the operation performed for each flow:

$$\Theta'_A(r_k) = \texttt{trust}((\Theta_A(r_k), W_k),\ (1 - A_{Ags}(\varphi_{i,j}), w_k)) \tag{2}$$

w_k is the weight of the update for the trustfulness associated with r_k, decreasing with the distance between $ix_A(\varphi_{i,j})$ and r_k:

$$w_k = e^{-dist_A(ix_A(\varphi_{i,j}), r_k)} \tag{3}$$

and W_k is the aggregated weight of all past updates to $\Theta(r_k)$. The operation `trust` denotes the weighted update of trustfulness, and depends on the trust model used in the mechanism. The trust model we use represents the trustfulness with triangular fuzzy numbers [20], with the core defined as average value of the trust observations (i.e. $1 - A_{Ags}(\varphi_{i,j})$), and the width of the fuzzy number representing the uncertainty or inconsistence of the past observations. The core of the fuzzy number (also denoted Θ_A in this paper, as it is used as a defuzzyfied value in the subsequent parts of the processing) is thus updated as:

$$\Theta'_A(r_k) = \frac{W_k \cdot \Theta_A(r_k) + w_k \cdot (1 - A_{Ags}(\varphi_{i,j}))}{W_k + w_k} \tag{4}$$

When we update the trust models with the first flow $\varphi_{1,1}$ of the first data set Φ_1, there are no centroids present yet, and the centroids are created progressively as the model processes the input data. Creation is based on a simplified Leader-Follower clustering algorithm [21], which always creates a new centroid with the same position as $ix_A(\varphi_{i,j})$ if a distance to the closest existing centroid is greater than predefined cutoff distance[1].

Collective trust estimation. In the last stage of processing, each agent determines the *trustfulness* $\Theta_A(\varphi_{i,j}\,\Phi_j)$ of each flow $\varphi_{i,j}$ from the current set Φ_j. To determine the trustfulness of individual flow $\varphi_{i,j}$, we aggregate the trustfulness $\Theta_A(r_k)$ associated with the centroids in the vicinity of flow's feature vector $ix_A(\varphi_{i,j})$. This operation is shown in Eq. 5.

$$\Theta_A(\varphi_{i,j}) = \texttt{weagg}_{r_k}(\Theta_A(r_k), w_k) \tag{5}$$

The operation `weagg` is a suitable weighted aggregation mechanism, and is determined by the trust model used for trustfulness representation. When we concretize the `weagg` operation as a weighted average, we obtain:

$$\Theta_A(\varphi_{i,j}) = \frac{\sum(\Theta_A(r_k)\cdot w_k)}{\sum w_k} \tag{6}$$

with both sums over the set of centroids in agent's trust model.

All agents provide their trustfulness assessment (which is conceptually a reputation opinion) for all flows to the aggregation and visualization agents, and the aggregated values are then used for traffic filtering. To perform the aggregation, we average the trustfulness opinions for each flow, as shown in Eq. 7. Aggregated trustfulness values for each flow constitute the output of the algorithm:

$$\Theta_{Ags}(\varphi_{i,j}) = \frac{1}{|Ags|}\sum_{X\in Ags}\Theta_X(\varphi_{i,j}). \tag{7}$$

3.3 Algorithm Properties

All three stages of the processing as listed above are designed to reach a joint conclusion between several anomaly detection method. We argue (and also show in the experiments on real networks) that the quality of joint conclusion is higher than the quality of the estimation provided by any single method, and that each of the algorithm stages contributes to quality improvement. In the following, we will discuss the elements that improve the quality of the conclusions reached by the proposed system.

[1] Cutoff distance thus determines the density of centroids in the feature space, and consequently the computational cost of the model. When the $ix_A(\varphi_{i,j})$ falls into the proximity of an existing centroid, the L-F algorithm specifies that the position of the closest centroid shall move in the feature space towards the last sample. In our implementation, such behavior would be highly undesirable (and thus is not implemented), as the values $\Theta_A(r_k)$ are relative to their positions, and any shift could impact their relevance.

Anomaly detection method integration. In Eq. 1, the algorithm integrates the anomaly values for each flow, and therefore minimizes the impact of traffic irregularities reported by a single method. This approach is not novel per se, but it already minimizes the impact of possible false positives using only the anomaly data, before the start of trust update process (see Fig. 2).

Trustfulness aggregation. During the update of the trust model, each agent creates the centroids r_k in its own feature space, and updates their trustfulness $\Theta_A(r_k)$ with the anomaly of flows in their vicinity. If there is only a **single agent** in the system (thus $A_{Ags}(\varphi_{i,j}) = A_A(\varphi_{i,j})$), this aggregation of anomalies into trustfulness has only limited impact. This is due to the fact that the features of the flow determine its feature vector $ix_A(\varphi_{i,j})$, and the anomaly $A_A(\varphi_{i,j})$ is determined from the vector values and status of the anomaly detection model. Therefore, we only aggregate the anomalies with the anomalies determined for similar flows in the past. However, even this aggregation helps to eliminate the effects of non-systematic irregularities in the traffic model.

When we combine **multiple agents** in the system, the situation becomes more interesting as each agents updates the anomaly value $A_{Ags}(\varphi_{i,j})$ over the centroids in their trust models. As each agent uses a distinct feature space and metrics, it has a different insight into the problem – the flows are positioned (clustered) according to the different criteria, and the cross correlation implemented by sharing of the anomaly values used to update the trustfulness helps to eliminate random anomalies. Let's assume that some of the flows constitute one anomaly, this anomaly was identified by a majority of agents and that the anomaly $A_{Ags}(\varphi_{i,j})$ of these flows is high. This implies that the flows are adjacent in the feature space of the agents that have identified them (because they are part of one anomaly identified by the anomaly detection model), and that during the trust update phase, their anomaly will significantly influence the trustfulness of one or few centroids r_k (see left segment of Fig. 3). On the other hand, when another agent does not detect these flows as anomalous, they will get dispersed among the clusters (not being recognized by agent's methods neither as anomalous, nor as similar to each other), and they will have only limited influence on the trustfulness of the centroids in their neighborhood (right segment of Fig. 3). This effect further eliminates the false positives, as it requires the untrusted flows to have previously unknown features, therefore creating a new centroid and pushing its trustfulness to low values, or to be similar to existing untrusted cluster(s).

From the computational complexity perspective, the trust update (and query) phases of the presented algorithm are characterized as $|\Phi_j| \cdot |\{r_k\}|$ for each agent [13], thus $|Ags| \cdot |\Phi_j| \cdot |\{r_k\}|$ for the whole system. It is linear in Φ_j, enabling the deployment of the system on the gigabit grade links using single multi-core PC. Memory requirements of each agent's model are linear in the number of centroids $|\{r_k\}|$. Complexity of individual anomaly detection algorithms is not considered in the estimation, but has not been a concern on real data.

4 Experimental Evaluation

In order to evaluate the effectiveness of our approach and to illustrate the properties discussed in Section 3.3, we have used the data acquired during the tests of our system on a live university network. We have performed two types of tests: in the first series, we have launched a series of scanning and profiling attacks against a selected host inside the surveyed network, and observed whether these attacks (using standard nmap scanner [22]) can be discovered on the background of the real traffic. In the second type of tests, we have tried to detect the third party attacks that were independently manually identified by network administrators.

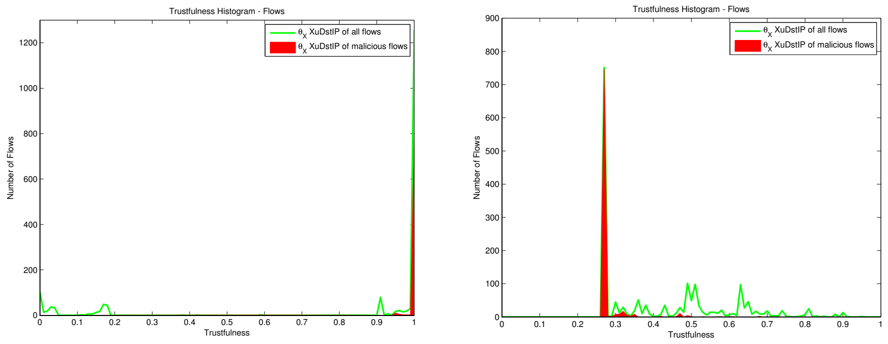


Fig. 2. Histogram of flow number over trustfulness from the results provided by **isolated** run of *XuDstIP* detection agent. (*left*) and the same agent when running with the others (*right*).

Before the presentation of aggregated results, we will present several situations that illustrate the algorithm behavior discussed in Section 3.3. In Figure 2, we can see the importance of anomaly aggregation as specified in Eq. 1 – when running alone, the agent is not able to detect the attack traffic (horizontal scan, 1000 flows, represented as surface in the graphs) as anomalous, and most of the attack flows are then classified with trustfulness close to 1. On the other hand, when the agent cooperates with the others and uses the common anomaly value $A_{Ags}(\varphi_{i,j})$, it is able to classify the traffic as untrusted. However, the classification is not perfect, as we can see from lower peaks of attack flow trustfulness distribution in the right segment of Fig. 2, between 0.3 and 0.5.

Perhaps the most important cooperative filtering effect is that the flows that are similar for one agent (and fall into the same feature space region) can be dispersed in the feature space of another agent – when they are coherent there as well, and are consistently untrusted by all agents, they are classified as an attack. We can illustrate these two situations in Fig. 3, where we project the attack and false positive flows respectively over the 3D projections of MINDS agent's centroids organized into the tree by similarity. We can see that the attack flows (from a vertical TCP SYN scan) are concentrated in a single centroid, and

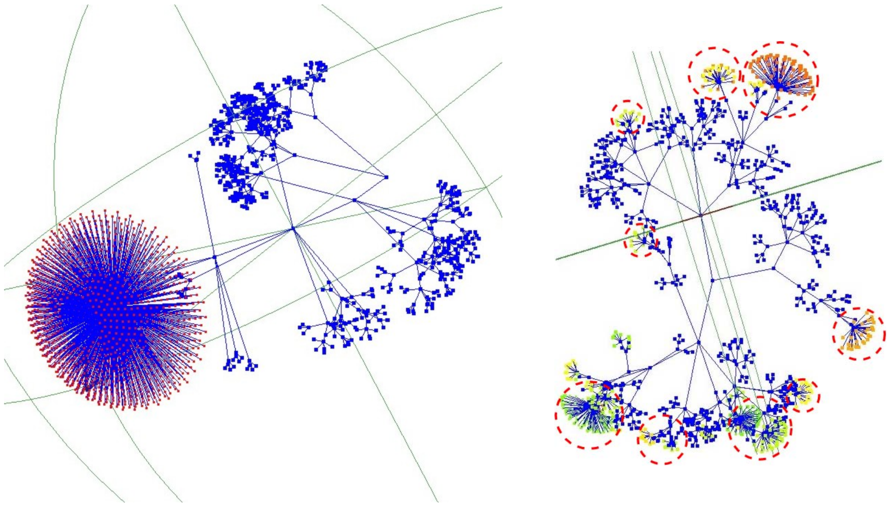


Fig. 3. 3D projection of agent's MINDS trust model. Centroids are organized in a tree, and each flow is attached to the closest centroid and colored by the centroid's trustfulness. The attack flows (*left*) tend to be concentrated around single centroid, while the false positives identified by another agent are spread over the whole model (*right*).

their high anomaly makes this newly created centroid untrusted. In the false positive case, the flows that were reported by another agent as a possible attack are dispersed over several existing centroids, and their higher anomaly value can not significantly influence the trustfulness of the centroids in their vicinity. Therefore, these flows will be reported with higher trustfulness.

In the experiments presented so far, we have seen the effects of the anomaly aggregation (Fig. 2) and projection on various feature spaces (Fig. 3). However, we should establish whether we need to perform the trust model update and query operations, instead of simply using the aggregated anomaly values directly. In a specific case of a small size vertical scan (300 UDP flows over 5 minutes), we can see that the aggregated anomaly of attack flows (red solid surface in the left segment of Fig. 4) is low, and they are not identified as malicious. On the other hand, the results of trustfulness aggregation provide much better results (Fig. 4, right), as they clearly classify the same traffic as untrusted. This is the effect of *a priori* low trustfulness associated with centroids around the attack flows in feature spaces of detection agents.

In Fig. 5, we can see the trustfulness assigned by individual agents and the whole system to the series of vertical scans that we have launched, in function of the number of flows during the observation period. The scans vary by approach (TCP SYN scan, UDP scan, profiling), port ordering and other parameters, but the results are relatively consistent. Larger scans are reliably detected, opinions of individual agents are relatively consistent, and the aggregation results detect all scans with the with more than 1 flow per second (averaged during the whole

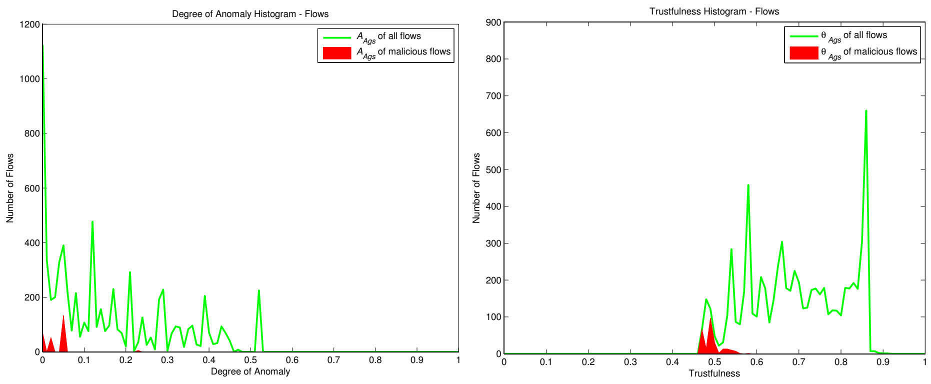


Fig. 4. Histogram of flow number by aggregated anomaly $A_{Ags}(\varphi_{i,j})$ (*left*). The attack (slow vertical scan, 300 flows, red surface) flows are not classified as anomalous. Histogram of flow number by aggregated trustfulness $\Theta_{Ags}(\varphi_{i,j}\,\Phi_j)$ (*right*) correctly classifies the attack as untrusted.

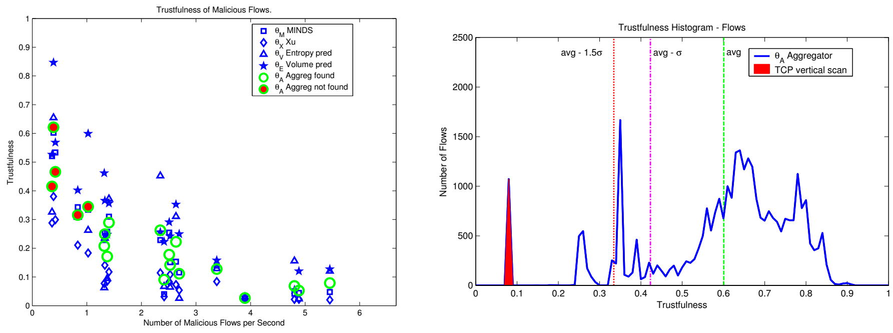


Fig. 5. Trustfulness assigned to vertical scans in function of attack intensity in the set Φ_j (*left*), with a specific example of one attack's classification in the trustfulness histogram (*right*)

period). In this experiment, we say that the attack was detected if the scan flows are classified between the *avg*-σ threshold, as shown in the right section of Fig. 5.

Once we have determined the system performance limits, we have also evaluated the system on a 30-minute snapshot of real-world traffic, including the activity of two zombie network nodes and one buffer overflow attack. Table 1 presents the performance of the system in terms of false positives/false negatives, evaluated for flows and distinct incident source IP addresses. We present the results for individual anomaly detection agents (MINDS: $A_{\mathcal{M}}$, Xu: $A_{\mathcal{X}}$, Entropy: $A_{\mathcal{E}}$ and Volume: $A_{\mathcal{V}}$, as defined in Section 3.1), aggregated anomalies $A_{\mathbb{M}}$ as defined by Eq. 1, trustfulness opinions of individual agents (Θ_M, Θ_X, Θ_V, Θ_E) and the final system output Θ_{Ags} defined by Eq. 7. We can see that even a simple aggregation of anomaly models provides far better results than any separate

Table 1. Benchmark of anomaly models, aggregated anomaly and partial and final trustfulness. Averaged over 6 data sets, each with 5 minutes of traffic with about 40 000 flows in each dataset.

		Anomalous					Untrusted				
		$A_{\mathcal{M}}$	$A_{\mathcal{X}}$	$A_{\mathcal{E}}$	$A_{\mathcal{V}}$	A_{Ags}	Θ_M	Θ_X	Θ_E	Θ_V	Θ_{Ags}
flows	detected	6653	3246	13541	12375	**9911**	9149	9975	10704	9518	**9741**
	TP	35	168	5841	5868	**4709**	5242	5712	5833	5864	**5769**
	FP	6618	3078	7700	6507	**5202**	3907	4263	4872	3654	**3972**
	FP[%] all	15.9%	7.4%	18.5%	15.6%	**12.5 %**	9.4%	10.2%	11.7%	8.8%	**9.5%**
srcIP	detected	72.5	322.3	17.2	16.7	**12.5**	7.8	11.3	13.5	10.8	**6.7**
	TP	1.7	0.2	2.5	2.7	**2.3**	2.7	2.7	2.3	2.7	**2.7**
	FP	70.8	322.1	14.7	14.0	**10.2**	5.1	8.6	11.2	8.1	**4.0**
	FP[%] all	1.52%	6.94%	0.31%	0.30%	**0.22 %**	0.11%	0.19%	0.24%	0.18%	**0.09%**

model (low FP values for $A_{\mathcal{E}}$ and $A_{\mathcal{V}}$ are caused by the fact that both models only consider significant sources of traffic $\sim$ 10% of hosts). The use of trust modeling further improves the results by wide margin – we detect more attacks, with far less false positives. The difference is significant especially in number of detected traffic sources, where we have reduced the rate of false positives more than two fold compared to $A_{\mathbb{M}}$, while detecting the actual attacks more reliably. It shall be noted that the number of suspicious sources is a far better estimator of analysis effort, because of significant variability in incident size.

5 Related Work

The ideas presented in this paper are relevant to three artificial intelligence and computer science domains: trust modeling, pattern recognition and classification, and network intrusion detection. As we have already stated in Section 2, the trust models [10,7] are specialized knowledge structures that excel in their area of specialization: learning from past observations of trustees behavior, integrating the reputation opinion (and knowledge about social structures) received from other agents and inferring reliable trust value, which can be then used for trusting decisions. In the field of network intrusion detection, they provide a very compelling set of features: they consider trustfulness of reputation [12,23] or information sources [24], and integrate these methods with robust approaches to reputation integration [11,8]. Recent models also concentrate on context representation [14,13] or multidimensionality of trust [25].We currently use the trust model described in [20], which has an advantage of being iterative (i.e. does not hold the history of observations 1 - $A_{Ags}(\varphi_{i,j})$ for each r_k), and thus reduces the computational complexity of the solution.

The field of pattern recognition and classification [21] provides us not only with the formalism used to represent the traffic in the extended trust models, but directly addresses the problem of cooperative classification as such. In [26], the authors introduce a general framework that integrates a major part of previous work on classifier integration strategies, and this work is directly relevant to the last step of the algorithm introduced in Section 3. The integration of key concepts from the classification work into the trustfulness aggregation shall help us to further improve the system. Agent methods have already been suggested for similar purpose, in a cooperative person tracking domain [27].

The multi-agent approaches to intrusion detection problems are mostly used in host-based or hybrid host and network based IDS [28,29], which perform part of their sensing on the protected hosts. This allows them to detect several types of local malicious actions, such as suspicious API calls or anomalous application activity. In [28], Valeur *et al.* propose a general framework for alert correlation, which is able to integrate the data from several sensors, associate the activities that are part of single attack and distinguish typical sequences of attack actions. While our approach fits the general theoretical framework introduced in [28], it differs in many crucial aspects. All the sensors use different input data, integrated detection methods use XML-based IDMEF [30] as a common ontology, the events detected by individual methods are associated with each other using time windows, and the system puts a lot of emphasis on the detection of attack sequences specific to composite attacks. The method introduced in our work is based on the fact that all the agents use the same input data Φ_j and collaborate actively (by sharing anomalies) even before submitting their individual trustfulness values, which are subsequently combined.

6 Conclusion

This paper presents a very specific application of collaborative trust modeling in a highly competitive domain of network intrusion detection. The goal of our work is to improve the results provided by state-of-the-art, but still imperfect anomaly detection algorithms by an addition of overlay layer based on extended trust modeling and simple reputation mechanism. The method is able to robustly identify the significant network-level events (scans, Denial of Service attacks, worms, peer-to-peer networks) in the traffic and present them to human supervisor for inspection in a dedicated visualization agent [31].

The application of agent techniques to the problem of network behavior analysis showed that trust modeling techniques distributed over a dynamic community of detection agents can significantly improve the quality of results. Simple integration of anomaly detection algorithms reduces the error rate significantly, but most of the benefit is in the overlay trusting layer, which *reduces the rate of false positives and false negatives simultaneously* and therefore shows the added value that the trust modeling techniques can bring to the highly competitive field of network intrusion detection.

Acknowledgment. This material is based upon work supported by the European Research Office of the US Army under Contract No. N62558-07-C-0001 and W911NF-08-1-0250. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the European Research Office of the US Army. Also supported by Czech Ministry of Education grants 1M0567, 6840770038 (CTU) and 6383917201 (CESNET).

References

1. Scarfone, K., Mell, P.: Guide to intrusion detection and prevention systems (idps). Technical Report 800-94, NIST, US Dept. of Commerce (2007)
2. Denning, D.E.: An intrusion-detection model. IEEE Trans. Softw. Eng. 13, 222–232 (1987)
3. Cisco Systems: Cisco IOS NetFlow (2007), `http://www.cisco.com/go/netflow`
4. Čeleda, P., Kováčik, M., Koníř, T., Krmíček, V., Špringl, P., Žádník, M.: FlowMon Probe. Technical Report 31/2006, CESNET, z. s. p. o (2006), `http://www.cesnet.cz/doc/techzpravy/2006/flowmon-probe/`
5. Lazarevic, A., Ertöz, L., Kumar, V., Ozgur, A., Srivastava, J.: A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the Third SIAM International Conference on Data Mining (2003)
6. Bragg, R., Rhodes-Ousley, M., Strassberg, K.: Network Security; The Complete Reference. McGraw-Hill, New York (2004)
7. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artif. Intell. Rev. 24, 33–60 (2005)
8. Sabater, J., Sierra, C.: Reputation and social network analysis in multi-agent systems. In: Proceedings of AAMAS 2002, Bologna, Italy, pp. 475–482 (2002)
9. Ramchurn, S., Jennings, N., Sierra, C., Godo, L.: Devising a trust model for multi-agent interactions using confidence and reputation. Applied Artificial Intelligence 18, 833–852 (2004)
10. Castelfranchi, C., Falcone, R.: Principles of trust for mas: Cognitive anatomy, social importance, and quantification. In: Proceedings of the 3rd International Conference on Multi Agent Systems, p. 72. IEEE Computer Society Press, Los Alamitos (1998)
11. Josang, A., Gray, E., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems 4, 139–162 (2006)
12. Huynh, T.D., Jennings, N.R., Shadbolt, N.R.: An integrated trust and reputation model for open multi-agent systems. Journal of Autonomous Agents and Multi-Agent Systems 13, 119–154 (2006)
13. Rehak, M., Pechoucek, M.: Trust modeling with context representation and generalized identities. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676. Springer, Heidelberg (2007)
14. Rettinger, A., Nickles, M., Tresp, V.: Learning initial trust among interacting agents. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 313–327. Springer, Heidelberg (2007)
15. Ertoz, L., Eilertson, E., Lazarevic, A., Tan, P.N., Kumar, V., Srivastava, J., Dokas, P.: MINDS - Minnesota Intrusion Detection System. In: Next Generation Data Mining. MIT Press, Cambridge (2004)
16. Xu, K., Zhang, Z.L., Bhattacharryya, S.: Reducing Unwanted Traffic in a Backbone Network. In: USENIX Workshop on Steps to Reduce Unwanted Traffic in the Internet (SRUTI), Boston, MA (2005)
17. Lakhina, A., Crovella, M., Diot, C.: Diagnosis Network-Wide Traffic Anomalies. In: ACM SIGCOMM 2004, pp. 219–230. ACM Press, New York (2004)

18. Lakhina, A., Crovella, M., Diot, C.: Mining Anomalies using Traffic Feature Distributions. In: ACM SIGCOMM, August 2005, pp. 217–228. ACM Press, New York (2005)
19. Rehak, M., Pechoucek, M., Bartos, K., Grill, M., Celeda, P.: Network intrusion detection by means of community of trusting agents. In: IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2007 Main Conference Proceedings) (IAT 2007). IEEE Computer Society Press, Los Alamitos (2007)
20. Rehák, M., Foltýn, L., Pěchouček, M., Benda, P.: Trust Model for Open Ubiquitous Agent Systems. In: Intelligent Agent Technology, 2005 IEEE/WIC/ACM International Conference (2005); Number PR2416 in IEEE
21. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. John Wiley & Sons, New York (2001)
22. Lyon, G.: Nmap, `http://insecure.org/nmap/`
23. Yu, B., Singh, M.P.: Detecting deception in reputation management. In: AAMAS 2003, pp. 73–80. ACM Press, New York (2003)
24. Barber, K.S., Kim, J.: Belief revision process based on trust: Agents evaluating reputation of information sources. In: Falcone, R., Singh, M., Tan, Y.-H. (eds.) AA-WS 2000. LNCS (LNAI), vol. 2246, pp. 73–82. Springer, Heidelberg (2001)
25. Vu, L.-H., Aberer, K.: A probabilistic framework for decentralized management of trust and quality. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 328–342. Springer, Heidelberg (2007)
26. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20, 226–239 (1998)
27. Meshulam, R., Reches, S., Yarden, A., Kraus, S.: Mlbp: Mas for large-scale biometric pattern recognition. In: AAMAS 2006: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 1095–1097. ACM Press, New York (2006)
28. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A.: A comprehensive approach to intrusion detection alert correlation. IEEE Transactions on Dependable and Secure Computing 01, 146–169 (2004)
29. Shyu, M.L., Quirino, T., Xie, Z., Chen, S.C., Chang, L.: Network intrusion detection through adaptive sub-eigenspace modeling in multiagent systems. ACM Trans. Auton. Adapt. Syst. 2, 9 (2007)
30. IETF: RFC 4765:The Intrusion Detection Message Exchange Format (IDMEF), `http://tools.ietf.org/rfc/rfc4765.txt`
31. Rehak, M., Pechoucek, M., Celeda, P., Krmicek, V., Moninec, J., Dymacek, T., Medvigy, D.: High-performance agent system for intrusion detection in backbone networks. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676. Springer, Heidelberg (2007)

A Distributed Generative CSP Framework for Multi-site Product Configuration

Markus Zanker[1], Dietmar Jannach[2], Marius C. Silaghi[3], and Gerhard Friedrich[1]

[1] University Klagenfurt, Austria
{markus.zanker,gerhard.friedrich}@uni-klu.ac.at
[2] Technical University Dortmund, Germany
dietmar.jannach@cs.uni-dortmund.de
[3] Florida Institute of Technology (FIT), Melbourne, US
Marius.Silaghi@fit.edu

Abstract. Today's configuration systems are centralized and do not allow manufacturers to collaborate online for offer-generation or sales-configuration activities. However, the integration of configurable products into the supply-chain of a business requires the cooperation of the various manufacturers' configuration systems to jointly offer valuable solutions to customers. As a consequence, there is a need for methods that enable independent specialized agents to compute such configurations. Several approaches to *centralized* configuration are based on constraint satisfaction problem (CSP) solving. Most of them extend traditional CSP approaches in order to comply to the specific expressivity and dynamism requirements of configuration and similar synthesis tasks.

The distributed generative CSP (DisGCSP) framework proposed here builds on a CSP formalism that encompasses the *generative* aspect of variable creation and extensible domains of problem variables. It also builds on the distributed CSP (DisCSP) framework, supporting configuration tasks where knowledge is distributed over a set of agents. Notably, the notions of constraint and nogood are further generalized, adding an additional level of abstraction and extending inferences to types of variables. An example application of the new framework describes modifications to the ABT algorithms and furthermore our evaluation indicates that the DisGCSP framework is superior to classic DisCSP for typical configuration task problem encoding.

1 Introduction/Background

The paradigm of mass-customization allows customers to tailor (configure) a product or service according to their specific needs, i.e. the customer can select between several features that should be included in the configured product and can determine the physical component structure of the personalized product variant. Typically, there are technical and marketing restrictions on the valid parameter constellations and the physical layout. This has led manufacturers to develop methods for checking the feasibility of user requirements and for computing consistent solutions. Typically, this functionality is provided by product configuration systems (configurators), which have proven to be a successful application area for different AI techniques [18] such as description logics [11], or rule-based [2] and constraint-based solving algorithms. [5] describes the industrial use of constraint techniques for the configuration of large and complex systems such as telecommunication switches and [10] is an example of a powerful commercialised tool based on constraint satisfaction.

However, companies find themselves having to cooperate with other highly specialized solution providers as part of a dynamic coalition to offer customized solutions. The level of integration present in today's digital markets implies that software systems supporting selling and configuration tasks may no longer be conceived as standalone systems. A product configurator can be therefore seen as an agent with private knowledge that acts on behalf of its company and cooperates with other agents to solve a configuration task. This paper abstracts the *centralized* definition of a configuration task in [19] to a more general definition of a *generative* CSP that is also applicable to the wider range of synthesis problems. Furthermore, we propose a framework that allows us to extend DisCSPs to handle distributed configuration tasks by integrating the innovative aspects of local generative CSPs:

1. The constraints (and nogoods) are generalized such that they depend on the types rather than on the identities of variables. This also enables the following aspects to be treated more elegantly.
2. The number of variables of certain types that are active in the local CSP of an agent may vary depending on the state of the search process and is hence dynamic. In the DisCSP framework, the external variables existing within the system are predetermined.
3. The domain of the variables may vary dynamically. Some variables model possible connections and depend on the existence of components that could later be connected.

Importantly, we also describe the impact of the previously mentioned changes on asynchronous algorithms. In the following we motivate our approach with an example, Section 3 defines a generative CSP and in Section 4 a distributed generative CSP is formalized and presented together with extensions to current DisCSP frameworks. Finally, Section 5 evaluates DisGCSP encoding against classic DisCSP problem representation for typical configuration problems.

2 Motivating Example

The following presents a typical example problem from the domain of product configuration ([5]) where interconnected systems support plug-in modules and problem specific constraints describe the legal combinations of module types and their capacity as well as their associated parameters. Figure 1 depicts a problem where systems consist of modules of different types, namely A-, B-, and C-modules, and have optional connection points for these modules (denoted as ports). For reasons of presentation the example focuses only on a small subset of a larger configuration problem of a technical system (see dotted lines). System 1 consists of A- and B-modules where system 2 may have only A- and C-modules plugged in. The A-modules also act as an interface between the two systems, i.e. they are shared by them. In addition, a module of any type can be either set as active or inactive. The initial situation in Figure 1 depicts the customer specific requirement that the configuration result contains at least one A-module that is connected via a port to the sub-system. According to the compatibility restrictions that will be described in the following, the found solution includes two additional modules of type B and C, where all A- and C modules are set to active and all B-modules to inactive.

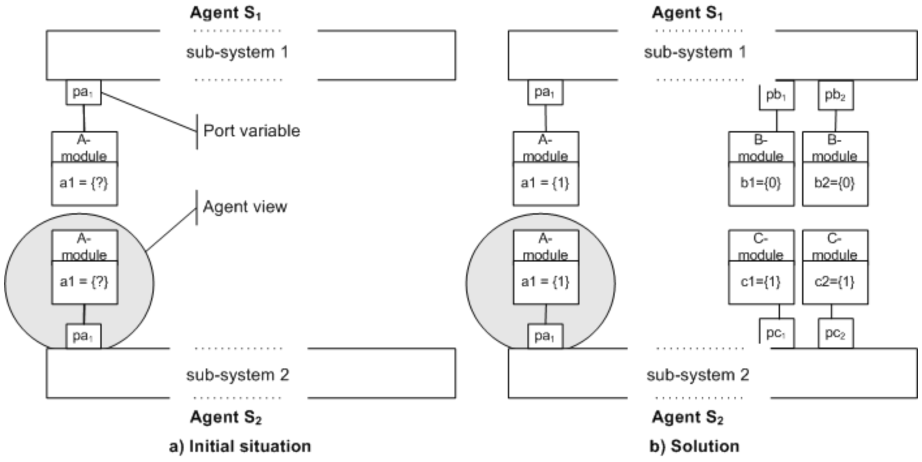


Fig. 1. Example problem

The distribution aspect is inherent in this scenario, as the overall solution consists of sub-systems that are to be configured by different agents. We formalize this configuration problem as a CSP, where each port and each module is represented by a variable[1]. Since the exact number of problem variables is not known from the beginning, constraints cannot be directly formulated on concrete variables. Instead, comparable to programming languages, variable types exist that allow to associate a newly created variable with a domain and we can specify relationships in terms of *generic constraints*. [19] define a generic constraint γ as a constraint schema, where meta-variables M_i act as placeholders for concrete variables of a specific type t, denoted by the predicate $type(M_i) = t$. The subscript i allows to distinguish between different meta-variables in one constraint[2]. In our example seven different types of problem variables exist,

[1] Note, that the *sub-system* components themselves are not explicitly modeled, but only via their characterizing port variables.

[2] The exact semantics of generic constraints is given in Definition 2 in Section 3.

representing the ports for the three different module types (t_{pa}, t_{pb}, t_{pc}) and the activation status for each type of the modules (t_a, t_b, t_c) as well as a type (t_{ct}) of counter variables (x_{type}) for the number of instantiations of each type. The configuration constraints are distributed between the agents, i.e., each agent S_i posesses a set of local constraints[3] Γ^{S_i}, i.e., $\Gamma^{S_1} = \{\gamma_1, \gamma_2, \gamma_5, \gamma_7, \gamma_9\}$ and $\Gamma^{S_2} = \{\gamma_3, \gamma_4, \gamma_6, \gamma_8, \gamma_{10}\}$, that are defined as follows:

Agent S_1 ensures, that the amount of B-modules and its associated port variables must not be above 3.
$\gamma_1 : val(x_{pb}) \leq 3.$ and $\gamma_2 : val(x_b) \leq 3.$ where $val(x)$ is a predicate that gives the assigned value of variable x.
Similarly for agent S_2, the amount of C-modules and its associated port variables must not be above 3.
$\gamma_3 : val(x_{pc}) \leq 3.$ and $\gamma_4 : val(x_c) \leq 3.$
Agent S_1 resp. S_2 check that there are more B- as well as C-modules than A-modules in a configured system.
$\gamma_5 : val(x_b) > val(x_a).$ and $\gamma_6 : val(x_c) > val(x_a).$
Agent S_1 resp. S_2 ensure that all B- resp. all C-modules have set the same activation status.
$\gamma_7 : type(M_1) = t_b \wedge type(M_2) = t_b \wedge val(M_1) = val(M_2).$ and
$\gamma_8 : type(M_1) = t_c \wedge type(M_2) = t_c \wedge val(M_1) = val(M_2).$
For agent S_1 A- and B-modules must not have the same activation status.
$\gamma_9 : type(M_1) = t_a \wedge type(M_2) = t_b \wedge val(M_1) \neq val(M_2).$
For agent S_2 A- and C-modules must have the same activation status.
$\gamma_{10} : type(M_1) = t_a \wedge type(M_2) = t_c \wedge val(M_1) = val(M_2).$

During the search process the search space is continuously extended by the instantiation of additional problem variables, until a solution is found that satisfies all the constraints of each agent. The *Agent view* contains the problem variables shared between agents in order to assure the evaluation of local constraints. In our case γ_6 and γ_{10} are so-called inter-agent constraints, that require agent S_2 to have access to all A-modules and its associated port variables.

Consequently, a solution to a generative constraint satisfaction problem requires not only finding valid assignments to variables, but also determining the exact size of the problem itself. In the sequel of the paper we define a model for the local configurators and we detail extensions to DisCSP algorithms.

3 Generative Constraint Satisfaction

In many applications, solving is a *generative* process, where the number of involved components (i.e., variables) is not known from the beginning. To represent these problems we employ an extended formalism that complies to the specifics of configuration and other synthesis tasks where problem variables representing

[3] In the example we omit those constraints that ensure that once a port variable is assigned a value, the corresponding connected component variable must exist.

components of the final system are *generated* dynamically as part of the solution process because their total number cannot be determined beforehand. The framework is called *generative* CSP (GCSP) [6,19]. This kind of dynamicity extends the approach of dynamic CSP (DCSP) formalized by Mittal and Falkenhainer [12], where all possibly involved variables are known from the beginning. This is needed because the activation constraints reason on the variable's activity state. [13] propose a conditional CSP to model a configuration task, where structural dependencies in the configuration model are exploited to trigger the activation of subproblems. Another class of DCSP was first introduced by [4] where constraints can be added or removed independently of the initial problem statement. The dynamicity occuring in a GCSP differentiates from the one described in [4] in the sense that a GCSP is extended in order to find a consistent solution and the latter has already a solution and is extended due to influence from the outside world (e.g., additional constraints) that necessitates finding a new solution. Here we give a definition of a GCSP that abstracts from the configuration task specific formulation in [19] and applies to the wider range of synthesis problems.

Definition 1 (Generative constraint satisfaction problem (GCSP)). *A generative constraint satisfaction problem is a tuple GCSP(X, Γ, T, Δ), where:*

- X *is the set of problem variables of the GCSP and* $X_0 \subseteq X$ *is the set of initially given variables.*
- Γ *is the set of generic constraints.*
- $T = \{t_1, \ldots, t_n\}$ *is the set of variable types* t_i*, where* $dom(t_i)$ *associates the same domain to each variable of type* t_i*, where the domain is a set of atomic values.*
- *For every type* $t_i \in T$ *exists a counter variable* $x_{t_i} \in X_0$ *that holds the number of variable instantiations for type* t_i*. Thus, explicit constraints involving the total number of variables of specific types and reasoning on the size of the CSP becomes possible.*
- Δ *is a total relation on* $X \times (T, N)$*, where* N *is the set of positive integer numbers. Each tuple* $(x, (t, i))$ *associates a variable* $x \in X$ *with a unique type* $t \in T$ *and an index* i*, that indicates* x *is the* i^{th} *variable of type* t*. The function* $type(x)$ *accesses* Δ *and returns the type* $t \in T$ *for* x *and the function* $index(x)$ *returns the index of* x*.*

By generating additional variables, a previously unsolvable CSP can become solvable, which is explained by the existence of variables that hold the number of variables.

When modeling a configuration problem, variables representing named connection points between components, i.e., *ports*, will have references to other components as their domain. Consequently, we need variables whose domain varies depending on the size of a set of specific variables [19].

Example. Given t_a as the type of variables representing *A-modules* and t_{pa} as the type of *port* variables that are allowed to connect to *A-modules*, then the domain of the *pa* variables $dom(t_{pa})$ must contain references to *A-modules.* This is specified by defining $dom(t_{pa}) = \{1, \ldots, ub\}$, where *ub* is an upperbound on the number of variables of type t_a, and formulating an additional generic constraint that restricts all variables of type t_{pa} using the counter variable for the total number of variables having type t_a, i.e., $type(M_1) = t_{pa} \wedge val(M_1) \leq x_{t_a}$. With the help of the *index*() function concrete variables can then be referenced. Referring to our introductory example we can formalize the local GCSP of agent S_1 in the initial situation (see Figure 1) as $X^{S_1} = \{x_a, x_{pa}, x_b, x_{pb}, x_{ct}, a_1, pa_1\}$, $\Gamma^{S_1} = \{\gamma_1, \gamma_2, \gamma_5, \gamma_7, \gamma_9\}$, $T^{S_1} = \{t_{ct}, t_a, t_{pa}, t_b, t_{pb}\}$ and $\Delta^{S_1} = \{(x_a, (t_{ct}, 1)), (x_{pa}, (t_{ct}, 2)), (x_b, (t_{ct}, 3)) (x_{pb}, (t_{ct}, 4)), (x_{ct}, (t_{ct}, 5)), (a_1, (t_a, 1)) (pa_1, (t_{pa}, 1))\}$. The $index(S_1)$ function returns 1, which indicates that a_1 is the first A-module instance. The domains of variables are consequently defined as $dom(t_a) = dom(t_{pa}) = dom(t_b) = dom(t_{pb}) = dom(t_{ct}) = \{1, \ldots, ub\}$, where the domains for the port variables are additionally limited by domain constraints (e.g., γ_1).

Definition 2 (Generic constraint). *A generic constraint $\gamma \in \Gamma$ formulates a restriction on the meta-variables $M_a, \ldots, M_k$. A meta-variable M_i is associated a variable type $type(M_i) \in T$ and must be interpreted as a placeholder for all concrete variables x_j, where $type(x_j) = type(M_i)$.*

Note, that generic constraints can also formulate restrictions on specific initial variables from X_0 by employing the *index*() function.

Consider the GCSP(X, Γ, T, Δ) and let $\gamma \in \Gamma$ restrict the meta-variables $M_a, \ldots, M_k$, where $type(M_i) \in T$ is the defined variable type of the meta variable M_i, then the consistency of generic constraints is defined as follows:

Definition 3 (Consistency of generic constraints). *Given an assignment tuple θ for the variables X, then γ is said to be satisfied under θ, iff $\forall x_a, \ldots, x_k \in X : type(x_a) = type(M_a) \wedge \ldots \wedge type(x_k) = type(M_k) \rightarrow \gamma[M_a|_{x_a}, \ldots, M_k|_{x_k}]$ is satisfied unter θ, where $M_i|_{x_i}$ indicates that the meta-variable M_i is substituted by the concrete variable x_i.*

Thus a *generic* constraint must be seen as a constraint scheme that is expanded into a set of constraints after a preprocessing step, where meta-variables are replaced by all possible combinations of concrete variables having the same type, e.g., given a fragment of a GCSP of agent S_1 (excluding counter and port variables) with $X^{S_1} = \{a_1, b_1, b_2\}$, $T^{S_1} = \{t_a, t_b\}$ and $\Delta^{S_1} = \{(a_1, (t_a, 1)), (b_1, (t_b, 1)), (b_2, (t_b, 2))\}$, the satisfiability of the generic constraint γ_9 is checked by testing the following conditions: $val(a_1) \neq val(b_1)$, $val(a_1) \neq val(b_2)$.

Definition 4 (Solution for a generative CSP). *Given a generative constraint satisfaction problem GCSP(X_0, Γ, T, Δ_0), then its solution encompasses the finding of a set of variables X, type and index assignments Δ and an assignment tuple θ for the variables in X, s.t.*

1. *for every variable $x \in X$ an assignment $x = v$ is contained in θ, s.t. $v \in dom(type(x))$ and*
2. *every constraint $\gamma \in \Gamma$ is satisfied under θ and*
3. *$X_0 \subseteq X \wedge \Delta_0 \subseteq \Delta$.*

Note, that we do not impose a minimality criterium on the number of variables in our solution, because in practical applications different optimization criteria exist, such as total cost or flexibility of the solution, thus non-minimal solutions can be preferred over minimal ones.

The calculated solution (excluding counter variables) for the local GCSP of agent a_1 consists of $X^{S_1} = \{a_1, pa_1, b_1, b_2, pb_1, pb_2\}$, $\Delta^{S_1} = \{(a_1, (t_a, 1))$, $(pa_1, (t_{pa}, 1))$, $(b_1, (t_b, 1))$, $(b_2, (t_b, 2))$,$(pb_1, (t_{pb}, 1))$,$(pb_2, (t_{pb}, 2))\}$ and the assignment tuple $a_1 = 1, pa_1 = 1, b_1 = 0, b_2 = 0, pb_1 = 1$ and $pb_2 = 2$. Thus, $b_1, \ldots, b_2$ and $pb_1, \ldots, pb_2$ are the names of *generated* variables.

Note, that names for generated variables are unique and can be randomly chosen by the GCSP solver implementation and therefore constraints must not formulate restrictions on the variable names of generated variables. Consequently, substitution of any generated variable (i.e., $x \in X \setminus X_0$) by a newly generated variable with equal type, index and value assignment has no effect on the consistency of generic constraints. Our GCSP definition extends the definition from [19] in the sense that a finite set of variable types T is given and during problem solving variables having any of these types can be generated, whereas in [19] only variables of a single type, i.e., component variables, can be created. Current CSP implementations of configuration systems (e.g., [10] [5]) use a type system for problem variables, where new variable instances, having one of the predefined types, are dynamically created. This is only indirectly reflected in the definition of [19] by the domain definition of component variables, which we explicity represent here as a set of types. Furthermore, the definition of *generic constraints* does not enforce the use of a specific constraint language for the formulation of restrictions. Examples are the LCON language used in the COCOS project [19], or the configuration language of the ILOG Configurator [10].

Note, that the set of variables X can be theoretically infinite, leading to an infinite search space. For practical reasons, solver implementations for a GCSP put a limit on the total number of problem variables to ensure decidability and finiteness of the search space. This way a GCSP is reduced to a dynamic CSP and in further consequence to a CSP. A DCSP models each search state as a static CSP, where complex activation constraints are required to ensure the alternate activation of variables depending on the search state. These constraints need to be formulated for every possible state of the GCSP, which leads to combinatorial explosion of concrete constraints. Furthermore, the formulation of large configuration problems as a DCSP is merely impractical from the perspective of knowledge representation, which is crucial for knowledge-based applications such as configuration systems.

4 DisGCSP Framework

Algorithms for configuration applications need to guarantee a good/optimal solution, that's why we focus on complete algorithms in our framework. The first asynchronous complete search algorithm is Asynchronous Backtracking (ABT) [21]. An enhanced version for several variables per agent is described in [22]. [3] shows how ABT can be adapted to networks where not all agents can directly communicate to one another. [7] makes the observation that versions of ABT with polynomial space complexity can be designed. Extensions of ABT with asynchronous maintenance of consistencies, and asynchronous dynamic reordering are described in [20,15,17]. [14] achieves an increased level of abstraction in DisCSPs by letting nogoods (i.e. certain constraints) consist of aggregates (i.e. sets of variable assignments), instead of simple assignments.

We show how the basic DisCSP framework for ABT [21] can be applied to a scenario of distributed product configuration. Therefore, improving the performance of ABT with extensions as referenced above is straightforward. We summarize in the following the properties of the ABT algorithm that guarantee its correctness and completeness [21]. Then we apply this DisCSP framework to a scenario where each agent locally solves a generative constraint satisfaction task. Each time an agent extends the solution space of his local GCSP by creating an additional variable, the DisCSP setting is transformed into a new DisCSP setting, which again has all properties required by asynchronous search to correctly function.

4.1 Asynchronous Search

We summarize the characteristics of asynchronous search algorithms like ABT [21], reformulated to allow agents to know only the constraints that they enforce. They are considered as follows:

1. $A = \{S_1, \ldots S_n\}$ is a set of n totally ordered agents (i.e. representing different sub-systems), where S_i has priority over S_j if $i < j$.
2. Each agent S_i owns a variable[4] and knows all the constraints that involve its variable and only variables of higher priority agents.[5] The constraints known by S_i are referred to as its local constraints, denoted Γ^{S_i} and S_i is *interested in* those variables that are contained in its local constraints. A *link* exists between two agents if they share a variable, that is directed from the agent with higher priority to the agent with lower priority. A link from agent S_1 to agent S_2 is referred to as an *outgoing link* of S_1 and an *incoming link* of S_2.
3. An assignment is a pair (x_j, v_j), where x_j is a variable, and v_j a value for x_j.
4. The *view* of an agent S_i is a set of the most recent assignments received for those variables agent S_i is interested in.
5. The agents communicate using the following types of messages, where channels without message loss are assumed:
   - **ok?** message. Agents with higher priorities communicate via each **ok?** message an assignment for their variable to lower priority agents.
   - **nogood** message. In case an agent cannot find assignments that do not violate its own constraints and its stored nogoods, it generates an

[4] As described later, one can see this variable as a tuple of variables treated simultaneously.

[5] In the original description of ABT, an agent also knows constraints on variables of higher priority agents.

explanation under the form of an explicit nogood $\neg N$. A nogood can be interpreted as a constraint that forbids a combination of value assignments to a set of variables. It is announced via a **nogood** message to the lowest priority agent that has proposed an assignment in N.
- **addlink** message. The receiver agent is informed that the sender is interested in its variable. A *link* is established from the higher priority agent to the agent with lower priority.

4.2 Framework for DisGCSP

A distributed configuration problem is a multi-agent scenario, where each agent wants to satisfy a local GCSP and agents keep their constraints private for security and privacy reasons, but share all variables which they are interested in. As constraints employ meta-variables, the *interest* of an agent in variables needs to be redefined:

Definition 5 (Interest in variables). *An agent S_j owning a local $GCSP^{S_j}(X^{S_j},\Gamma^{S_j},T^{S_j},\Delta^{S_j})$ is said to be interested in a variable $x \in X^{S_h}$ of an agent S_h, if there exists a generic constraint $\gamma \in \Gamma^{S_j}$ formulating a restriction on the meta-variables $M_a,\ldots,M_k$, where $type(M_i) \in T^{S_j}$ is the defined variable type of the meta variable M_i, and $\exists M_i \in M_a,\ldots,M_k : type(x) = type(M_i)$.*

Definition 6 (Distributed generative CSP). *A distributed generative constraint satisfaction problem has the following characteristics:*

- *$A = \{S_1,\ldots,S_n\}$ is a set of n agents, where each agent S_i owns a local $GCSP^{S_i}$ $(X^{S_i}, \Gamma^{S_i}, T^{S_i}, \Delta^{S_i})$.*
- *All variables in $\bigcup_{i=1}^{n} X^{S_i}$ and all type denominators in $\bigcup_{i=1}^{n} T^{S_i}$ share a common namespace, ensuring that a symbol denotes the same variable, resp. the same type, with every agent.*
- *For every pair of agents $S_i, S_j \in A$ and for every variable $x \in X^{S_j}$, where agent S_i is interested in x, must hold $x \in X^{S_i}$.*
- *For every pair of agents $S_i, S_j \in A$ and for every shared variable $x \in X^{S_i} \cap X^{S_j}$ the same type and index must be associated to x in the local GCSPs of the agents, i.e., $type^{S_i}(x) = type^{S_j}(x) \wedge index^{S_i}(x) = index^{S_j}(x)$.*

Consequently, for every pair of agents $S_i, S_j \in A$ and for every shared variable $x \in X^{S_i} \cap X^{S_j}$ a *link* must exist that indicates that they share variable x. The *link* must be directed from the agent with higher priority to the agent with lower priority.

Definition 7. *Given a distributed generative constraint satisfaction problem among a set of n agents then its solution encompasses the finding of a set of variables $X = \bigcup_{i=1}^{n} X^{S_i}$, type and index assignments $\Delta = \bigcup_{i=1}^{n} \Delta^{S_i}$ and an assignment tuple $\theta = \bigcup_{i=1}^{n} \theta^{S_i}$ for every variable in X, s.t. for all agents $S_i : X^{S_i}$, Δ^{S_i} and θ^{S_i} are a solution for the local $GCSP^{S_i}$ of agent S_i.*

Remark. A solution to a distributed generative CSP is also a solution to a centralized GCSP$(\bigcup_{i=1}^{n} X^{S_i}, \bigcup_{i=1}^{n} \Gamma^{S_i}, \bigcup_{i=1}^{n} T^{S_i}, \bigcup_{i=1}^{n} \Delta^{S_i})$.

Definition 8 (Generic assignment). *A generic assignment is a unary generic constraint. It takes the form: $\langle M, i, v \rangle$, where M is a meta-variable, i is a set of index values for which the constraint applies, and v is a value.*

Definition 9 (Generic nogood). *A generic nogood takes the form $\neg N$, where N is a set of generic assignments for distinct meta-variables.*

Value assignments to variables are communicated to agents via **ok?** messages that transport *generic assignments* in our DisGCSP framework, which represent domain restrictions on variables by unary constraints. Each of these unary constraints in our DisGCSP has attached an unique identifier called constraint reference (cr) [16]. Any inference has to attach the crs associated to arguments into the obtained nogood. We treat the extension of the domains of the variables as a constraint relaxation [16]. For this reason we introduce the next features for algorithm extensions:

- **announce** message broadcasts a tuple (x, t, i), where x is a newly created variable of type t and with index i to all other agents. The receiving agents determine their interest in variable x and react depending on their interest and priority in one of the following ways (a) send an **addlink** message transporting the variable set $\{x\}$ (b) add the sending agent to its outgoing links or (c) discard the message.
- **domain** message broadcasts a set CR of obsolete constraint references. Any receiving agent removes all the nogoods having attached to them a constraint reference $cr \in CR$. The receiver of the message calls then the function *check_agent_view()* detailed in [21], making sure that it has a consistent proposal or that it generates nogoods.
- **nogood** messages transport *generic nogoods* $\neg N$ that contain assignments for meta-variable instances. These messages are multicasted to all agents interested in $\neg N$.[6] An agent S_i is interested in a generic nogood $\neg N$ if it has *interest in* any meta-variable in $\neg N$.
- When an agent needs to revoke the creation of a new variable due to backtracking in his local solving algorithm, he assigns it a specific value from its domain indicating the deactivation of the variable and communicates it via an **ok?** message to all interested agents.

In order to avoid too many messages a broker agent can be introduced that maintains a static list of agents and their interest in variables of specific types comparable to a *yellow pages* service. In this case the agent that created a new variables only needs to request the broker agent for a list of interested agents and does not need to broadcast an **announce** message to all agents.

Theorem 1. *Whenever an existing extension of ABT is extended with the previous messages and is applied to DisGCSPs, the obtained protocols are correct, complete and terminate.*

[6] The algorithm remains correct and terminates even if the nogoods are sent only to the target decided as in ABT.

Proof: Let us consider that we extend a protocol called P.

Completeness: All the generated information results by inference. If failure is inferred (when no new component is available), then indeed no solution exists.

Termination: Without introducing new variables, the algorithm terminates. Since the number of variables that can be generated is finite, termination is ensured.

Correctness: The resulting overall protocol is an instance of P, where the delays of the system agent initializing the search equals the time needed to insert all the variables generated before termination. Therefore the result satisfies all the agents and the solution is correct.

5 Evaluation

In order to test the applicability of our approach, we implemented a prototype for distributed generative constraint satisfaction on top of ILOG's *JConfigurator* [8]. *JConfigurator* is a Java library providing an API for modeling and solving configuration problems based on an underlying object-oriented constraint solver. Consistent with the GCSP approach, the user of this library defines the problem in terms of components, ports and attributes and states generic constraints that apply to the set of all instances of a specific component type [8,10].

Our framework provides a simple, experimental infrastructure for distributed reasoning among an arbitrary number of agents, each of which is capable of solving a local GCSP, where there are no limitations on the number of component types or the complexity of the constraints for the local GCSPs. However, for the purposes of evaluating the framework, despite the lack of benchmarking problems, we restricted the *structure* of the configuration problem to being similar to the example in Section 2 which still captures the main characteristics of configuration problems. Our tests were limited to ports that connected the sub-systems with the modules and component types that were characterized solely by one integer attribute (with a finite numerical domain). However, the maximum number of the sub-system ports and component instances were only limited to a theoretical value.

In order to be able to compare our DisGCSP framework with the conventional DisCSP framework, the example configuration problems were also modelled as static CSPs with all possible component instances being generated prior to execution, where their domain was extended to include an additional value indicating their inactivity. Thus, we were able to examine the effect of defining nogoods and constraints *generically* on the number of interaction cycles between agents and compare it with the classical constraint and nogood formulation in DisCSPs. In addition, we found that the additional computational costs for deriving minimal conflicts pays off given the potentially high communication overhead of the message passing associated with additional interaction cycles.

Architecture. The framework's core is an *Agent* class that manages the *agent view* and implements a variant of Yokoo's Asynchronous Backtracking algorithm (i.e., sending and processing **ok?** and **nogood** messages). Concrete agent instances (with their local problems) are implemented by subclassing and overriding application specific methods, for example, the definition of components and constraints. Communication among agents is based on message passing via a *mediating agent* that provides capabilities for agent registration and system initialization and is capable of detecting when the distributed system reaches a stable state, i.e., a solution is found.

Conflict detection and exchange. The computation of nogoods (minimal conflicts) in the case that the agent view is inconsistent, is based on Junker's QUICKXPLAIN algorithm [9], a efficient non-intrusive conflict-detector that recursively partitions the problem into subproblems of half the size and skips those that do not contain an element of the propagation-specific conflict[7]. In the current version, we only compute a single minimal, conflict in each backtracking step, future work will include the concurrent computation of several conflicts. A computed conflict contains information about the number of variables involved in the conflict as well as inconsistent variable assignments, where we can detect when the inconsistency arises solely from variable cardinalities which further improves the distributed search performance. Conflict exchange among agents is based on serialization of the conflict information and the receiving agent's automated (re-)construction of the generic constraint.

Algorithm. Given the results from previous sections, several distributed constraint satisfaction algorithms can be employed to solve the distributed configuration problem. In our framework we currently employ a variant of Yokoo's sound and complete ABT algorithm without employing enhancements like dynamic agent ordering [1] or Aggregation Search [14]. This choice was mainly driven by the characteristics of the configuration domain, where the order of the agents is mainly determined by the supply chain setting. The extensions include the handling of multiple variables by aggregating variables according to their types: agents can request links to variable *types* (component types in configuration terminology); thus, **ok?** messages contain the assignments of all currently existing variables of a given type, where each variable type is owned by exactly one agent. The computation of local solutions is performed by the underlying constraint solver. The additional task of applying dynamic agent ordering or configuration-specific heuristics remains part of our future work.

Measurements. Several initial tests (Table 1) were carried out on our framework using the configuration problems as described above, varying the size of the configuration problem, the number of agents as well as the local search strategies and problem complexity in order to obtain significant distributed search and backtracking activity[8]. The results shown in Table 1 present the behavior of the various distributed systems for the same problem ecoded both as

[7] Note, that we are only interested in propagation-specific conflicts that are induced by the values in the agent view.

[8] Note, that in the configuration domain, the number of co-operating agents of the companies involved in the supply chain is typically very low (< 10), the agents are usually loosely coupled and the problems are typically underconstrained.

Table 1. Comparison of DisGCSP and DisCSP encoding

Encoding	Nbr. of agents	Nbr. of CT	Nbr. of inst	Shared inst	Overall time	Check-time	NG	Msgs	Checks
DisGCSP(1)	3	10	30	12	3.25	1.72	**25**	**75**	134
DisCSP(1)					23.20	14.30	**165**	**477**	820
DisGCSP(2)	6	22	120	27	8.53	3.21	**40**	**126**	210
DisCSP(2)					37.47	14.28	**211**	**644**	1105
DisGCSP(3)	10	30	140	65	13.90	5.05	**63**	**205**	375
DisCSP(3)					89.60	34.50	**594**	**1792**	3024
DisGCSP(4a)	12	36	164	87	15.32	5.03	**66**	**238**	403
DisCSP(4a)					127.44	38.30	**705**	**2635**	4094
DisGCSP(4b)	12	36	164	87	41.02	8.30	**136**	**588**	889
DisCSP(4b)					2600	128.00	**3646**	**16476**	23452

CT: component/variable types
inst: instances
shared instances: shared component instances
Check-time: consistency, search and explanation per agent
NG: overall number of recorded nogoods/backtracks
Msgs: overall number of messages
Checks: overall number of consistency checks and searches

Generative Constraint Satisfaction Problem with a given upper bound of possible component instances and as a static CSP. For distributed reasoning, the identical variant of Yokoo's ABT search (with support for multiple variables per agent) is employed, where in the case of the GCSP problem *generic* nogoods are exchanged among the agents. In both settings the explanation facilities for computing minimal nogoods were utilized.

It is well known that formalisms that extend the static CSP paradigm such as Dynamic CSP or Generative CSP have advantages for non-distributed problem solving both from modeling, knowledge acquisition, and maintenance perspectives as well as from a solution search point of view. In a distributed settings where the configuration constraints are distributed among several cooperating agents, the non-generic approach suffers from the problem of heavy messaging traffic that is induced by the increased number of required interaction cycles for finding a solution. In the case of traditional CSP encoding, a receiving agent is only capable of computing minimal conflicts involving concrete variable instances, however in the generic case the agent can deduce and report *generic* nogoods to the sending agent. It is the fact that - in the GCSP and configuration problem setting - individual variable (i.e., component) instances of a given type are interchangeable. Therefore, reporting a nogood prevents the sending agent from communicating an interchangeable solution which would again cause an inconsistency for the receiving agent.

Table 1 contains the average time measurements for finding the first solution in five different configuration scenarios with varying complexity. Each problem instance was examined several times as differences occur due to the indeterministic behavior of the parallel execution of the agents. The problem sizes (i.e. the number of component types or agents) are realistic for the scenarios addressed within the CAWICOMS project. Furthermore, the scenarios reflect the fact that in a supply chain setting only a small portion of the local configuration problems are shared among the agents. The local GCSPs are underconstrained; using more complex problems would result in an increase in the amount of time needed for consistency checks and the local solution search which is done by the constraint solver. The actual number of problem variables is determined by the number of component instances, the cardinality variables for all types and the internally generated variables that allow the formulation of n-ary constraints. While the net search times are secondary[9], the experiments showed that the generative variant performs significantly better in terms of required interaction cycles, stored nogoods and messages.

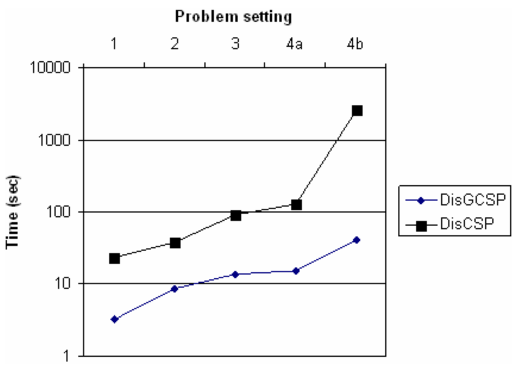


Fig. 2. Comparing run times for different problem settings

Figure 2 visualizes the run times for the different sample problems. Note that problem instances 4(a) and 4(b) are identical in terms of problem size and distribution among agents but differ in search complexity, i.e., problem 4(b) contains a problematic constraint constellation that causes the run-times of the static CSP approach to increase dramatically. While message passing is quite cheap in our multi-threaded prototype, the cost of agent communication in real distributed environments is a crucial factor. Memory requirements for storing nogoods are not problematic because they are minimal and can hence be represented in a compact way; however, minimizing the number of search cycles by reducing the search space through the elimination of interchangeable solutions leads to overall performance enhancements especially in cases where the local configuration

[9] The time measurements where made on a standard PC where the parallel agent threads run in one single process; overall memory consumption was in all DisGCSP test cases below 25 MB.

problems are complex. Beside the advantage of offering a faster distributed solution search, the GCSP approach has significant advantages for the domain of real-world distributed configuration problems in terms of knowledge maintenance: the problem of modeling and maintaining shared agent knowledge and agent interdependencies is neglected in many DisCSP approaches and alleviated in a DisGCSP setting through the introduction of variable types and generic constraints, thus eliminating the need for error-prone task of encoding problems as static CSPs.

Finally, the experiments showed that the integration of distributed configuration capabilities into a commercial configuration tool like *JConfigurator* is feasible and lays the foundation for the application of distributed constraint solving in real-world environments.

6 Conclusions

Building on the definition of a centralized configuration task from [19], we formally defined a new class of CSP, termed generative CSP (GCSP), that generalizes the approaches of current constraint-based configurator applications [5,10]. The innovative aspects include an additional level of abstraction for constraints and nogoods. Constraints and nogoods may consist of variable types instead of solely variables. Furthermore, we extended GCSP to a distributed scenario, allowing DisCSP frameworks to be adapted to dynamic configuration problems (but it can be used in static models as well) and described how this enhancement can be integrated into a large family of existing asynchronous DisCSP algorithms. Initial evaluations indicate that GCSP is practical for typical distributed configuration problems.

References

1. Armstrong, A., Durfee, E.F.: Dynamic prioritization of complex agents in distributed constraint satisfaction problems. In: Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI), Nagoya, Japan (1997)
2. Barker, V.E., O'Connor, D.E., Bachant, J.D., Soloway, E.: Expert systems for configuration at Digital: XCON and beyond. Communications of the ACM 32(3), 298–318 (1989)
3. Bessière, C., Maestre, A., Meseguer, P.: Distributed dynamic backtracking. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, p. 772. Springer, Heidelberg (2001)
4. Dechter, R., Dechter, A.: Belief Maintenance in Dynamic Constraint Networks. In: Proc. 7th National Conf. on Artificial Intelligence (AAAI), St. Paul, MN, pp. 37–42 (1988)
5. Fleischanderl, G., Friedrich, G., Haselböck, A., Schreiner, H., Stumptner, M.: Configuring Large Systems Using Generative Constraint Satisfaction. In: Freuder, E., Faltings, B. (eds.) IEEE Intelligent Systems, Special Issue on Configuration, vol. 13(4), pp. 59–68 (1998)
6. Haselböck, A.: Knowledge-based configuration and advanced constraint technologies. PhD thesis, Technische Universität Wien (1993)

7. Havens, W.: Nogood caching for multiagent backtrack search. In: Proc. of 14th National Conf. on Artificial Intelligence (AAAI), Agents Workshop, Providence, Rhode Island (1997)
8. Junker, U.: Preference-based programming for Configuration. In: Proc. of IJCAI 2001 Workshop on Configuration, Seattle, WA (2001)
9. Junker, U.: QuickXPlain: Conflict Detection for Arbitrary Constraint Propagation Algorithms. In: Proc. of IJCAI 2001 Workshop on Modelling and Solving problems with constraint, Seattle, WA (2001)
10. Mailharro, D.: A classification and constraint-based framework for configuration. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(4), 383–397 (1998)
11. McGuiness, D.L., Wright, J.R.: Conceptual Modeling for Configuration: A Description Logic-based Approach. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(4), 333–344 (1998)
12. Mittal, S., Falkenhainer, B.: Dynamic Constraint Satisfaction Problems. In: Proc. of 8th National Conf. on Artificial Intelligence (AAAI), Boston, MA, pp. 25–32 (1990)
13. Sabin, D., Freuder, E.C.: Configuration as Composite Constraint Satisfaction. In: Proc. of AAAI Fall Symposium on Configuration, AAAI Press, Cambridge (1996)
14. Silaghi, M.-C., Sam-Haroud, D., Faltings, B.: Asynchronous search with aggregations. In: Proc. of 17th National Conf. on Artificial Intelligence (AAAI), Austin, TX, pp. 917–922 (2000)
15. Silaghi, M.-C., Sam-Haroud, D., Faltings, B.: ABT with asynchronous reordering. In: Proc. of Intelligent Agent Technology (IAT), Maebashi, Japan, pp. 54–63 (October 2001)
16. Silaghi, M.-C., Sam-Haroud, D., Faltings, B.V.: Maintaining hierarchically distributed consistency. In: Proc. of 7th Int. Conf. on Principles and Practice of Constraint Programming (CP), DCS Workshop, Singapore, pp. 15–24 (2000)
17. Silaghi, M.-C., Sam-Haroud, D., Faltings, B.V.: Consistency maintenance for ABT. In: Walsh, T. (ed.) CP 2001. LNCS, vol. 2239, pp. 271–285. Springer, Heidelberg (2001)
18. Stumptner, M.: An overview of knowledge-based configuration. AI Communications 10(2) (June 1997)
19. Stumptner, M., Friedrich, G., Haselböck, A.: Generative constraint-based configuration. Artificial Intelligence for Engineering Design, Analysis and Manufacturing 12(4), 307–320 (1998)
20. Yokoo, M.: Asynchronous weak-commitment search for solving large-scale distributed constraint satisfaction problems. In: Proc. of 1st Int. Conf. on Multi-Agent Sytstems (ICMAS), San Francisco, CA, pp. 318–467 (1995)
21. Yokoo, M., Durfee, E.H., Ishida, T., Kuwabara, K.: Distributed constraint satisfaction for formalizing distributed problem solving. In: Proc. of 12th Int. Conf. on Distributed Computing Systems (ICDCS), Yokohama, Japan, pp. 614–621 (1992)
22. Yokoo, M., Hirayama, K.: Distributed constraint satisfaction algorithm for complex local problems. In: Proc. of the 3rd Int. Conf. on Multi-Agent Systems (ICMAS), Paris, France, pp. 372–379 (1998)

MobiSoft: Networked Personal Assistants for Mobile Users in Everyday Life

Christian Erfurth[1], Steffen Kern[1], Wilhelm Rossak[1],
Peter Braun[2], and Antje Leßmann[3]

[1] Friedrich Schiller University Jena, Computer Science Department
Ernst-Abbe-Platz 2, 07743 Jena, Germany
{erfurth,kern,rossak}@informatik.uni-jena.de
[2] The agent factory GmbH
c/o Intershop Tower, Leutragraben 1, 07743 Jena, Germany
braun@the-agent-factory.de
[3] Godyo AG
Prüssingstraße 35, 07745 Jena, Germany
antje.lessmann@godyo.com

Abstract. This paper provides an overview of the MobiSoft project, it's ideas and aims as well as the achieved results. In MobiSoft, we applied mobile software agents to support humans in their mobile everyday life. We developed a generic application framework that can be customized to fit into completely different scenarios ranging from industry use cases to social human interactions during leisure time. We describe this framework as well as several prototypes that demonstrate its general applicability. This paper also delivers first results of a survey at the university campus, that tried to capture user interest in personal assistants and mobile applications in general.

1 Introduction to MobiSoft

Over the last years, mobile devices, in particular mobile phones, have become part of our daily life and our indispensable companions. They help us managing our appointments, contact lists, or personal tasks. It can be expected that they will become even more powerful and widespread in the near future. They will –or already do– provide support for various wireless network technologies that allow for the establishment of personal area networks (PAN) in order to exchange information with others in close proximity as well as to access information stored on distant hosts across the Internet.

MobiSoft is an ongoing shared venture between Friedrich Schiller University Jena (FSU), the agent factory GmbH, and Godyo AG and is funded by the Thuringian Ministry of Economy, Technology and Labor. In MobiSoft we aim at different application scenarios that range from information retrieval and control of legacy systems in industry use cases to the support of human interactions in all places where people come together.

Mobile employees in varous industries are the central user group in the first type of applications. We aimed at supporting such mobile users with mobile software agents that are able to discover other agents in the network and exchange information efficiently and that are also capable to discover and use services to solve given tasks. This works transparently for the user who has only stated a request on a mobile device. Later, the agents on the mobile devices inform their respective users about the results of their actions and let them decide on further steps. By this, we overcome existing boundaries like tedious, manual browsing by delegating such tasks to software agents. Concerning these application types we had the following key aims:

– Access to critical information anytime and everywhere. Assistants can filter such data on company servers and only crucial information is delivered as soon as the user connects to a network to allow in-time reaction to important news.
– Access and control of business processes by utilizing several mobile personal assistants which act proactive and autonomously on their owners behalf.

The second type of application scenarios share the goal to initialize and ease social interactions. Our trageted users are, as before, humans which travel around and meet at specific places, for example shopping malls, sports stadiums, public transport, museums, libraries, conferences, lecture halls, etc. Although it might be helpful and interesting, people rarely start talking to complete strangers, because of inhibitions, social barriers or simply a lack of time. Otherwise, if people knew each other, they would more freely exchange information and, therefore, spread and receive pieces of useful information that could further be combined with already existing information and forwarded to others.

We aim at supporting such a very human behavior of information exchange with mobile software agents that act as user representatives and reside on mobile devices. We assume that users have delegated the task of finding proper human communication partners to these agents. We conceive software agents to be small entities that are situated in a networked environment of mobile devices. Agents are able to react to their virtual environment; in our case this environment is made up of other agents in proximity. The goal is to establish communication between the local personal assitant and the other agents according to the needs of the given scenario. Communication between agents is based on messages, which are annotated with semantic information as defined in an *agent communication language* (ACL) using high-level communication protocols such as negotiations [37]. For more information about software agents in general, we refer to [39].

Our approach can be seen as complementary to more traditional techniques for information discovery on mobile devices. At first, instead of a client-server based communication, we propose a decentralized, peer-to-peer like technique to handle the dynamics and complexity of mobile networks. Second, the process of searching for and dissemination of information is proactively performed by the personal assistant rather than by the human user. Finally, the goal of our approach is on the one hand the establishment of social interactions, mainly in consumer scenarios, and on the other hand information retrieval and distribution. We therefore utilize assistants that are able to migrate through dynamic networks.

In this project, we address several research problems that are at the intersection of distributed computing, mobile ad hoc networks, and information representation using standardized languages and ontologies. We are aware of several additional research issues, for example in the area of privacy protection and human-computer interaction to make this type of application both useful and widely acceptable for users. We see this project as a first step in which we aim at developing the framework and technical infrastructure that will also enable later studies of those issues in detail.

The rest of this paper is structured as follows: The next chapter introduces three application scenarios that have been elaborated during the project. Afterwards, we introduce the application framework and describe certain aspects and technologies in detail. Section 4 covers related work followed by a short report on a survey covering the acceptance of mobile applications which we conducted at FSU. The paper closes with an summary of the MobiSoft project and a short look at further research challenges.

2 Application Scenarios

Within MobiSoft, we investigated different application scenarios which we ranked as realistic for future usage in mobile contexts. The scenarios, namely *Project Assistant*, *Social-mobile Assistant*, and *Campus.NET*, address different aspects of technology application. During the project, Campus.NET and Social-Mobile Assistant scenarios developed strong community aspects. Despite the fact that these scenarios have fewer business aspects, they received more attention during presentations of the project results, especially at Centrum der Büro- und Informationstechnik (CeBIT) 2007. This section will introduce all three scenarios.

2.1 Project Assistant

In this scenario a project manager is supported by personal electronic assistants which act autonomously and proactive on their owners behalf. Running on mobile devices, assistants can filter information on company servers and only crucial information is delivered as soon as the user connects to a network, fast enough to provide for in-time reaction to important news. This enables the manager to access critical business information anytime and everywhere as well as to access and control business processes by utilizing several personal assistants.

Imagine the following use case. You are a supply chain manager in a company which delivers products for several different industries like automotive or iron and steel industry. Your products are integrated into industry-specific and time-critical production workflows. Additionally, your company itself has numerous sub-suppliers, which are integrated into your own production.

In this position you are at the intersection of several business workflows acting as a negotiator for different interests. You work out new contracts with business partners, exchange old contractors with new ones, alter your production workflows or control your company's stock. Any information about delays in those well-defined workflows or changes in any contract or your own production line is highly important. Any of these events could slow down or ultimately stop your production, reduce your profit and may in the end lead to a lost contract or customer.

This use case exhibits two general problems. First, the supply chain manager will be often away from his office traveling to and meeting partners. During this time he has no access to critical information and thus no chance of intervention. Second, many tasks, which are handled by the supply chain manager, are tedious and error-prone. Additionally, the number of different tasks is continuously rising increasing workload and introducing new points of failure.

We try to address them both. First, we want to provide a supply chain manager with all necessary information at any time anywhere by making that data accessible with a mobile device like PDA or mobile phone. Second, we want to decrease the workload by assigning most of the routine and tedious tasks to software assistants which will act autonomously and proactively. Those assistant will only bother their owner if it is absolutely necessary. There will not only be assistants for the supply chain manager but also assistants, which represent contractors or customers. So the assistants may interact with each other to handle tasks previously performed by humans. More over, those assistant will be capable to handle some tasks, like negotiating on new contracts, faster than humans. The complexity of the system will be hidden behind a user interface which will only display the crucial information and aspects of the systems state.

For more details on this scenario please have a look at [21].

2.2 Social-Mobile Assistant

Focus of this scenario are humans and their direct interaction. With the help of their networked mobile devices, social-mobile assistants exchange information in order to find matches for their interests. If found, assistants inform their owners. Such social-mobile Assistants can be applied for

– The establishment of groups based on shared interests (work, hobbies) or activities and goals (such as to reduce travel costs by sharing a taxi).
– The exchange of information, such as personal profiles, news, private sales, or any kind of recommendations.
– The preselection of possible communication partners in social networks and the coordination of shared task lists and diaries by automated negotiations.

Our *social* approach can be seen as complementary to more traditional techniques for information discovery on mobile devices. The goal is mainly the establishment of social interactions rather than just information distribution. A decentralized peer-to-peer technique which is based on the notion of proximity is used.

The process of searching for and disseminating information is pro-actively initiated by the personal assistant rather than by the human user. Fig. 1 shows

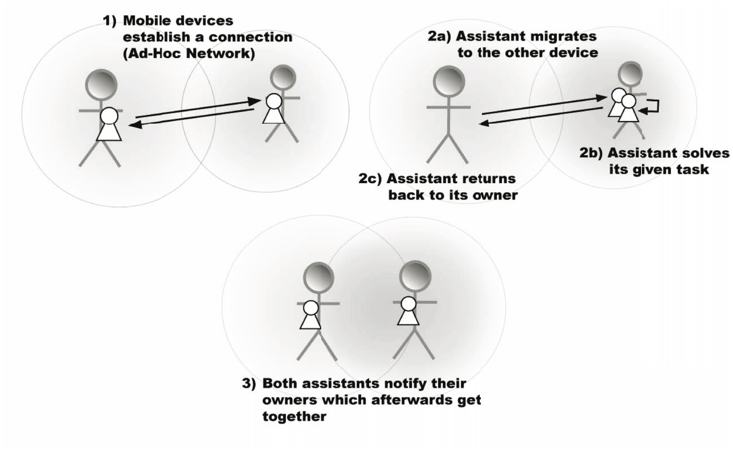

1) Mobile devices establish a connection (Ad-Hoc Network)
2a) Assistant migrates to the other device
2b) Assistant solves its given task
2c) Assistant returns back to its owner
3) Both assistants notify their owners which afterwards get together

Fig. 1. General Execution Steps during Face-to-Face Encounter of two mobile Users

this process. This information exchange works transparently for the user only in the first steps, in which the assistants exchange information, such as user profiles, or negotiate best interaction time. Later, the assistants inform their respective users about the potential communication partner and let them decide on further steps. By this, we overcome existing inhibitory behavior of humans by delegating this task to software agents, while the agents' goal is to find *proper* communication partners and *interesting* information. In [22], we cover the social-mobile aspects of MobiSoft in more detail.

2.3 Campus.NET

As the name Campus.NET already indicates this scenario targets issues of academic life. Typically, at a university there are different information systems in various contexts, like enrollment, library, eLearning, administration etc. Some of these heterogeneous systems might be integrated for higher convenience. Mobile access and push services are useful add-ons.

On the one hand, for students as well as university staff a lot of information is available for their support but distributed over these systems. On the other hand, with teaching or studying in a special field users get a certain view on available information and only a sub-set is of interest. Especially for a beginner, it is difficult to access necessary information and to get an overview of available support. Additionally there are only a few tasks and deadlines one needs to observe. Campus.NET is intended to be firstly a portal to heterogeneous information sources, secondly a watch dog for interesting events and finally a community platform to find helpful or interesting contacts with other people. This scenario is therefore the most complex one of those introduced. Within MobiSoft Campus.NET takes advantage of the well developed infrastructure at FSU and the commitment of students and university staff to support our ideas.

3 Architecture and Technology

In this section, we will outline the technological foundations of the project as well as the developments made during the project. As stated above, our experiences and research result on agent systems and mobile agents that have been aquired over many years and provided a sound basis to start with [8,6,14,7].

At first we are going to describe the second version of our *Agent Toolkit Tracy* followed by the introduction of a special Tracy version for Java-enabled mobile devices called *TracyME.* Along with the development of TracyME, we established a new agent programming language – called *TAL* – to overcome the limitations of mobile Java editions and to allow for agent miration between mobile devices. During the project, we also implemented an administation UI based on the Eclipse Framework to allow for easy usage and configuration of numerous Tracy agencies. This UI will be described in short, too.

Second, we will describe a number of extentions to our Tracy base system, so-called plugins, that provide additional services und necessary functionality, like a peer-to-peer network overlay, automatic integration of Web Services into Tracy, or the monitoring of agent networks.

Afterwards, we present several prototypes of the current system that show its general applicability and the wide range of possible use cases.

3.1 TracySE, TracyME and TAL

Over the past years, we've conducted a lot of fundamental research on agent systems and, especially, mobile agents. During those years, we've developed two Agent System Toolkits named Tracy and Tracy2. The latter one naturally benefitted from our experiences with the first version. The major difference is, that Tracy2 relies on a micro kernel architecture, compare Figure 2, whereas its predecessor applied a strict 3-tier design which proved to be far to monolithic in action. On top of the kernel, a number of *plugins* provide the actual functionality of a Tracy system. For example, message exchange among agents, agent migration or several security mechanism are realized as plugins. The last, missing part to make up a complete agent system are the domain-specific agents. As with plugins, the kernel is responsible to control agent lifecycles and to coordinate interactions among agents and plugins. Over the last years, this highly modular and extensible architecture proved to be useful, as it is very easy to adapat a base Tracy system to custom needs by defining a set of necessary plugins and corresponding agents.

One of the main MobiSoft aims was to port Tracy2 to the Java Micro Edition environment to allow for the usage of agents and agent migration on mobile devices. Other projects, like JadeLeap [3], use mobile devices merely as a remote control for agents that reside on a distant host. Our goal was to actually host agents on mobile devices and allow agent migration between mobile phones via Bluetooth and to distant hosts over GPRS, UMTS or WiFi.

Moving Tracy2 to a J2ME environment proved to be merely straightforward. At first, we ported the micro kernel to have the base execution environemt for

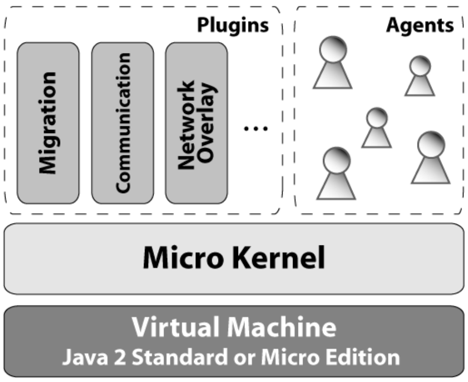


Fig. 2. General architecture of TracySE and TracyME

agents and plugins running on mobile devices. Afterwards, we began to transfer necessary plugins. Some of them only required minor adaptions whereas others had to be completely rewritten. Furthermore, we implemented several new plugins, i.e. for Bluetooth communication and user profile handling. Of all plugins, that have been reimplemented for the Java 2 Micro Edition, the most challenging one was the migration plugin. Agent migration, i.e. an agents movement from one host to another host, relies on several programming language features, that are not available in J2ME. Namely, class loader and dynamic class loading as well as object serialization. To overcome these limitations, we established a new, java-like programming language, that is interpreted by a virtual machine running under J2ME. The name of this language is simply *TAL* (The Agent Language).

With this solution, we are able to run and migrate agents between two and more mobile devices using Bluetooth or WiFi. Even the fact that TAL is a new languge is not a great drawback as its syntax is very similar to Java and implementation of agents rather straightforward.

The new Tracy system for mobile Java editions is called *TracyME* (Tracy Micro Editon) and, along with this, we renamed Tracy2 to *TracySE* (Tracy Standard Edition).

3.2 Network Types and Communication Techniques

In Mobisoft, we targeted different network types and applied different techniques to support and handle these networks. On the one end, we got fixed, hard-wired networks with normal workstations and servers and, on the other end, we looked into true ad hoc networks, established by mobile devices alone. In between these two extremes are networks that we call *semi* or *managed ad hoc*. These networks consist of a static, hard-wired core that provides a kind of backbone for numerous mobile devices which are connected with each other via Bluetooth or WiFi and, by means of GPRS/UMTS or WiFi, are connected to the central core. Our main focus has been on ad hoc networks. Nevertheless, we aimed for solutions that could be used for static networks, too.

To structure static and semi ad hoc networks we applied a peer-to-peer approach using JXTA [1]. First, JXTA is available for various platforms including J2SE and J2ME environments. Second, [9] presented a powerful peer-to-peer routing mechanism for mobile devices in ad hoc networks which is implemented in JXTA. First, we use JXTA to publish information about available Tracy platforms. Second, JXTA is used to publish information about available services. This allows agents and platforms to find desired functionality or information dynamically and independent of a specific platform.

Nevertheless, the choice for JXTA should not lead to a tight coupling between Tracy and JXTA, thus we aimed at providing an abstract network management plugin which is just a wrapper for a network overlay. The plugin provides a fixed interface for other plugins and agents to find Tracy platforms and search for respectively publish platform and service information. The plugin will delegate those requests to a network overlay module –in our case JXTA– which is capable to provide these services. Using this kind of delegation and loose coupling, it is easy to replace JXTA with one or more better alternatives without changing any agent or plugin that relies on the network management.

In full ad hoc networks, a complex overlay that relies on rendevous server (as in case of JXTA) is not applicable. Thus, we headed for more lightweight techniques like broadcasts or epidemic dissemination [20,36,5] to distribute agents and information among participating devices. Preliminary test results show the principal applicability of such algorithms. However, reliability and speed of distribution need to be increased to live up to the requirements of real applications.

Besides integrating network management for platform and service propagation and discovery, we also implemented a network monitor plugin. With this plugin, we are able to control and measure a network of Tracy platforms, get information for each platform, running agents and their tours, and currently connected users. This plugin helped us in our system tests to find bottlenecks and to pinpoint problems during in each stage of the project. Moreover, our adminstration UI *Wai Lin* greatly benefits from the data collected by this plugin.

3.3 Prototypes

In the course of the project, we implemented several prototypes to verify our current approaches and ideas as well as to derive new directions for reasearch based on the prototypes' performance. In general, we had tree different prototypes in parallel, each one covering one of the scenarios described in section 2.

The Project Assistant prototype delivers a base architecture and appropriate agents to oversee and manipulate an enterprise resource planning (ERP) system from a mobile device. The prototypes architecture is made up from the following parts: the ERP system P/4 [15], a TracySE system connected to P/4 via numerous Web Services and corresponding wrapper plugins, and several mobile devices running TracyME and agents that are able to migrate to the TracySE system and access the aforementioned wrapper plugins to manipulate the ERP system.

These agents are able to monitor certain critical values like availabe stock or contract conditions. In case of anything critical, they will migrate to their users mobile device and deliver the necessary information. The user may now take the appropriate actions by contacting other company members or making changes in the system – both using agents.

The second prototype covers social-mobile scenarios. Here, each user carries a personal assistant on his/her mobile device that maintains the users profile and searches for other assistants in the vicinity using Bluetooth. We presented the first version of the prototype at the CeBIT exhibition in 2006 and were highly disappointed by its poor performance. The incredible amount of Bluetooth-enabled devices at the fair and the rather faulty Bluetooth implementation on various mobile phones have been the main reason for these bad results. For CeBIT 2007, we successfully reimplemented our TracyME Bluetooth plugin to avoid all shortcomings of the various Bluetooth implementations on our test phones. The prototype has been much more stable and reliabile – nearby phones hosting matching profiles nearly always contacted each other and exchanged information.

For the Campus.NET scenario, we implemented several assistants each one covering a specific use case of the scenario. For example, we got one assistant that is able to access the library (which hosts a TracySE platform) and get information on books in general or that can monitor a books status (available, conferred, lost, ...) over a longer period. Another assistant covers the lecture timetable doing similar things as the library assistant, e.g. getting general information on courses or monitor a certain course or lecture. We also used this prototype to evaluate the simple dissemination of information, e.g. the menu of the cafeteria, among a number of mobile devices.

4 Related Work

Current approaches for mobile social applications [33] are based on central servers. For example, Dodgeball [13] and Playtxt [31] are social mobile networks to locate friends, friends of mutual acquaintances or other people with matching profiles. In those applications a user has to provide his or her current location manually, whereas in the Reno system [33] the current location is determined via GSM technology.

In this project we try to combine the advantages of both existing approaches, while avoiding a centralized architecture. On the one hand, we continue to use the concept of *places* rather than *locations*. A *place* is a logical description of an area such as *soccer stadium*, whereas a *location* is given by exact coordinates or cells. On the other hand, it is essential for our approach that proximity of humans can be determined transparently for users.

Therefore, we focus on applications that are based on mobile ad hoc networks (MANETs). A MANET is a collection of mobile devices (nodes), which can communicate with each other over a wireless network, for example WiFi or Bluetooth. By definition, mobile ad hoc networks have no fixed infrastructure, that is, all typical network functions need to be coordinated by the network nodes in a distributed manner. While WiFi is more suited for high-end mobile devices and for scenarios which demand long range ad hoc networks, Bluetooth is a promising communication technology for short range ad hoc networks like personal area networks (PAN) on which we focus in this project. A personal area network can be seen as a digital space around a person, whose size depends on the underlying wireless transmission technique. If two digital spaces overlap, people can virtually see each other, that is, their mobile devices are able to exchange information. The concept of MANET is very appealing both for research and industry, as it enables a new class of application that has not been possible so far. In a PAN we do not need an explicit notion of places and provision of proximity information is an inherent network function.

Most research into mobile ad hoc networks has focused on the problem of multi-hop routing data packets to enable Internet-like applications in ad hoc networks. In particular, they address the issues of how to enable peer-to-peer like applications on mobile devices [29] to share files [11], MP3 play-lists [38], or information dissemination of homogeneous data with one application such as traffic information [30]. Most of those approaches were mainly for enabling information exchange triggered manually by users [38] or make information dissemination completely independent of the user [30]. In contrast, our project aims at the establishment of social interactions by use of MANET.

In our project, we identify and focus on two key problems, which are related to the two main goals of our project as mentioned previously:

At first, we consider different strategies for information exchange [16]. *Flooding techniques* are based on the concept of broadcasting information units to all available nodes in the network. In general, flooding is a very simple technique that can be considered to be not appropriate in large networks, because of high resource usage in terms of bandwidth and energy consumption. The *publish/subscribe architectures*, well known from the Internet and also used for mobile applications connected to central servers on the Internet, are useful if the server (broker) can predict what type of information may be useful for the client. The more accurate this predication is the less data is sent superfluously over the network [40]. Profiles and application dependent requests are used to describe the client's needs and desires. Although, publish/subscribe architectures are very attractive, major requirements of this model, such as orderedness, consistency, and completeness make it difficult, if not impossible, to realise it in mobile networks. For example [10] [17] propose a publish/subscribe model for peer-to-peer and mobile networks without addressing the three requirements mentioned previously. *Epidemic dissemination* sends each information unit to a randomly chosen group of nodes. This dissemination approach enables messages to propagate quickly in the network and it is very robust against the node and network link failures. This approach works completely decentralised and must be seen in contrast to IP multicast techniques in which a spanning tree has to be set up from the source to all receiver nodes. In those techniques node or network link failures result in loss of messages, whereas in the epidemic-based approaches the messages can be delivered to a node via multiple (redundant) paths. For more information on those algorithms we refer to [20,36,5]. So far, the epidemic-based algorithms have only been studied as a general replacement for traditional routing and multicast algorithms in mobile ad hoc networks. Finally, *proximity-based algorithms* send information units only to neighbours, that is, other nodes in close proximity. At the moment it is not clear which of these approaches works best under which circumstances. Early results are only based on simulations [28] of small networks and focus on performance metrics rather than qualitative comparisons of the approaches.

The second aspect of our project deals with the problem of information representation in open environments which are not specific to a single application domain as existing approaches [30]. We aim to apply information dissemination approaches to distribute information about users, user interests and similar information. To make this approach flexible, extendable, and to base the matchmaking process between user interests and roaming information units, it is necessary to use semantically rich languages. Although there has been a lot of research done in the area of matchmaking of user interests and profiles [23] and the creation of social groups, today's available techniques are still quite simple. For example, in the Internet we find Tribe [34] that is a Web site enabling people to find other people based on their interest. The users must create a profile, can publish recommendations for restaurants etc., and establish or join tribes (online communities) dedicated to a specific topic. The description of user interests is based on a list of keywords. For mobile devices, we find Upoc [35] that can be used to establish communities on the Web and then send short messages to all members of a group or a content channel. These approaches compare user interests by comparing keywords. Jambo [19] provides a software for WiFi-enabled mobile devices so that users can locate each other based on keyword-based profile matching. Neither approach uses semantic descriptions of user profiles and preferences, but only simple text-based approaches. The first two approaches mentioned are based on central servers in the Internet, whereas the last approach requires a WiFi managed network. Other projects such as Webhound/Webdoggie [32] and HOMR/Ringo/Firefly [25] use similar approaches. Friend of a Friend Finder (FOAF) is a project that aims to share information about persons in the Internet. The language used in this project is RDF that provides a means to describe data and meta-data. RDF defines a simple data model which consists of resources and statements that link two resources, comparable to a subject-verb-object relationship. A statement is called a triple, which consists of subject and object, and the predicate that plays a role of the verb mentioned previously. In a so-called FOAF file, a user describes his personal data and which other persons this user knows using RDF. With a help of these links to other persons, a search engine can now create a graph of who knows whom. However it is not possible to describe user interests and preferences with FOAF.

We align ourself with [2] in that mobile agents roaming in mobile networks need to be light-weighted to work on battery operated and resource constraint devices like PDAs or mobile phones. Further, agents must be able to use different communication techniques and act context-specific.

A general concern in all mobile agent systems is security. An agent, that carries sensitive information about its owner must be protected against malicious agents or platforms. A discussion of security techniques lies beyond the scope of this paper. Please refer to [26,4] for an introduction into security issues in mobile agent systems and wireless networks.

In the area of supply chain management multi-agent systems are an alternative to common centralized client/server based systems. Multi-Agent Systems have proven to perform well in the areas of task scheduling and distribution, negotiation or process planing and control and they are already used in industry. See for example DISPOWEB[12], KRASH[24] or IntaPS[18] or [27]. With MobiSoft, we are not challenging those systems but we aim to make those systems accessible for our agents.

5 Lessons Learned

During the project we faced some technical difficulties in implementing our prototypes, e. g. inadequat API support for available communication interfaces, additional implementation effort due to different display resolutions as well as other device specific obstacles and restictions of Java 2 Micro Edition in gerneral.

More non-technical aspects in the social area of potential users are also relevant e. g.

– Has usability reached an appropriate level for ad hoc mobile device usage?
– Is there a correlation between age and gender of potential users and the acceptance of mobile application usage?
– Is the usage of mobile devices as a community interface accepted?

Consolidated findings in this interdisciplinary area are essential for a successful application of future technologies in the mobile sector. The user is the central element in ubiquitous computing.

Based on the results and prototypes in the Campus.NET scenario, we made a survey at the university to investigate the acceptance of personal electronic assistants. We received over 1000 submissions of the online questionary (70% from students, 23% from staff).

We ask the participants to rank importance of new services using electronical assistants and readiness of use. For 70% of participants, services with software assistants in the library sector are important or very important. In contrast to that, 61% of participants stated that it is less important or not important to have assistants for menu information on mobile devices. Sozial-mobile assistants are generally ranked as partly important.

The readiness of use is ranked nearly identical for all of these areas. Despite of the fact that the new technology of mobile personal assistants is not yet popular, about 50% of the interviewed persons answered that they would perhaps use assistants. Especially participants younger than 30 years are more interested in usage than the older ones which mostly answered *in no case*. So, the technology has the potential to get accepted in future.

6 Conclusions

Within MobiSoft we investigate the application of mobile software agents as personalized assistants on mobile devices, especially UMTS-enabled, in future application scenarios. The introduced scenarios adress different usage possibilities from business application to social interaction. Thereby, different network constellations from semi-static to ad hoc networked mobile devices are covered by the developed framework. With the prototypes we evaluate technology application with its challenges in the field of mobile applications. With our survey we get a first feedback of the technology acceptance from potential users.

MobiSoft has also shown that additional research and development effort has to be put into, e. g. semantical description of information for filtering and matching purposes, involvement of humans via mobile devices in dynamical business workflows, and improvement of usability of mobile devices for efficient information presentation and content creation.

Acknowledgments

The work presented in this paper is partially funded by the Thuringian Ministery of Economy, Technology and Labor under grant FKZ B 509-04005. Many thanks to Michael Selle, Judith Zimmermann, Jens Kramer, Norbert Wabnitz, Daniela Fiedler, Torsten Dettborn, Volkmar Schau, Kasten Jahn, our many students and all the others which contributed to MobiSoft actively.

References

1. JXTA., http://www.jxta.org
2. Bagci, F., Petzold, J., Trumler, W., Ungerer, T.: Ubiquitous mobile agent system in a p2p-network. In: UbiSys-Workshop at the Fifth Annual Conference on Ubiquitous Computing, Seattle, USA, October 12-15 (2003)
3. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing Multi-Agent Systems with JADE. Wiley & Sons, Chichester (2007)
4. Binder, W., Roth, V.: Secure mobile agent systems using java: Where are we heading (2002)
5. Birman, K.P., Hayden, M., Ozkasap, O., Xian, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Transaction on Computer Systems 17(2), 41–88 (1999)
6. Braun, P.: The Migration Process of Mobile Agents–Implementation, Classification, and Optimization. PhD thesis, Friedrich-Schiller-Universität Jena, Computer Science Department (May 2003)
7. Braun, P., Müller, I., Schlegel, T., Kern, S., Schau, V., Rossak, W.: Tracy: An Extensible Plugin-Oriented Software Architecture for Mobile Agent Toolkits. Whitestein Series in Software Agent Technologies, pp. 357–382. Birkhäuser, Basel (2005)
8. Braun, P., Rossak, W.R.: Mobile Agents–Basic Concept, Mobility Models, and the Tracy Toolkit. Morgan Kaufmann Publishers, San Francisco (2005)

9. Buccafurri, F., Lax, G.: Tls: A tree-based dht lookup service for highly dynamic networks. In: Meersman, R., Tari, Z. (eds.) OTM 2004. LNCS, vol. 3290, pp. 563–580. Springer, Heidelberg (2004)
10. Castro, M., Druschel, P., Kermarrec, A.-M., Rowstron, A.: Scribe: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications 20(8) (2002)
11. Christoph Lindemann, O.P.W.: A distributed search service for peer-to-peer file sharing in mobile applications. In: Proceedings fo the Second International Conference on Peer-to-Peer Computing (P2P 2002). IEEE Computer Society Press, Los Alamitos (2002)
12. DISPOWEB, `http://www.dispoweb.de`
13. Dodgeball, `http://www.dodgeball.com`
14. Erfurth, C.: Proaktive autonome Navigation für mobile Agenten. PhD thesis, Friedrich-Schiller-Universität Jena, Fakultät für Mathematik und Informatik (2004)
15. ERP-System P/4, Godyo AG, `http://www.godyo.com`
16. Franklin, M.J., Zdonik, S.B.: Dissemination-based information systems. Data Engineering Bulletin 19(3), 20–30 (1996)
17. Huang, Y., Garcia-Molina, H.: Publish/subscribe in a mobile environment. In: Banerjee, S. (ed.) Proceedings of the 2nd ACM International Workshop on Data Engineering for wireless and mobile access, Santa Barbara CA (USA), pp. 27–34. ACM Press, New York (2001)
18. IntaPS, `http://www.intaps.org`
19. Jambo, `http://www.jambo.org`
20. Kermarrec, A.-M., Massoulie, L., Ganesh, A.J.: Probabilistic reliable dissemination in large-scale systems. IEEE Transaction on Parallel and Distributed Systems 14(3), 248–258 (2003)
21. Kern, S., Braun, P., Dettborn, T., Eckhaus, R., Ji, Y., Erfurth, C., Rossak, W.: Assistant-based mobile supply chain management. In: Riebisch, M., Tabeling, P., Zorn, W. (eds.) 13th Annual IEEE International Symposium and Workshops on the Engineering of Computer Based Systems - Mastering the Complexity of Computer-based Systems (ECBS 2006), Potsdam (Germany), March 2006, pp. 23–31. IEEE, Los Alamitos (2006)
22. Kern, S., Braun, P., Rossak, W.: Mobisoft: An agent-based middleware for social-mobile applications. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 984–993. Springer, Heidelberg (2006)
23. Kleemann, T., Sinner, A., von Hessling, A.: Semantic user profiles and their applications in a mobile environment. In: Workshop on Artificial Intelligence in Mobile Systems at UbiComp 2004, Nottingham (UK) (September 2004)
24. KRASH, `http://www.ipd.uka.de/krash`
25. Lashkari, Y., Metral, M., Meas, P.: Collaborative interface agents. In: Proceedings of the 12th National Conference on Artificial Intelligence, Seattle (USA), vol. 1, pp. 444–449. MIT Press, Cambridge (1994)
26. Mavridis, I., Pangalos, G.: Security issues in a mobile computing paradigm (1997)
27. MultiAgent.com, `http://www.multiagent.com`
28. Nittel, S., Duckham, M., Kulik, L.: Information dissemination in mobile ad-hoc geosensor networks. In: Egenhofer, M.J., Freksa, C., Miller, H.J. (eds.) GIScience 2004. LNCS, vol. 3234. Springer, Heidelberg (2004)

29. Oberender, J., Andersen, F.U., de Meer, H., Dedinski, I., Hossfeld, T., Kappler, C., Maeder, A., Tutschku, K.: Enabling mobile peer-to-peer networking. In: Kotsis, G., Spaniol, O. (eds.) Euro-NGI 2004. LNCS, vol. 3427, pp. 219–234. Springer, Heidelberg (2005)
30. Ouri Wolfson, A.P.S., Xu, B.: An economic model for resource exchange in mobile peer-to-peer networks. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM 2004). IEEE Computer Society Press, Los Alamitos (2004)
31. Playtxt, http://www.playtxt.net
32. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating words of mouth. In: Proceedings of the Conference on Human Factors in Computing Systems. ACM Press, New York (1995)
33. Smith, I.: Social-mobile applications. Computer 38(4), 84–85 (2005)
34. Tribe, http://www.tribe.net
35. Upoc, http://www.upoc.com
36. Vogels, W., van Renesse, R., Birman, K.: The power of epidemics: robust communication for large-scale distributed systems. ACM SIGCOMM Computer Communication Review 33(1), 131–135 (2003)
37. Weiss, G. (ed.): Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press, Cambridge (2000)
38. Wiberg, M.: Folkmusic - a mobile peer-to-peer entertainment system. In: Proceedings of the 37th Hawaii International Conference on System Sciences. IEEE Computer Society Press, Los Alamitos (2004)
39. Wooldridge, M.: An Introduction to MultiAgent Systems. John Wiley and Sons, Chichester (2002)
40. Yoneki, E., Bacon, J.: An adaptive approach to content-based subscription in mobile ad-hoc networks. In: Proceedings of the 2nd IEEE Annual Conference on Pervasive Computing and Communication Workshops (PERCOMM 2004). IEEE Computer Society Press, Los Alamitos (2004)

A Web-Based Virtual Machine for Developing Computational Societies[⋆]

Sergio Saugar and Juan M. Serrano

Department of Computing
University Rey Juan Carlos
{Sergio.Saugar,JuanManuel.Serrano}@urjc.es

Abstract. Different theoretical and practical insights into the field of computational organisations and electronic institutions has led to a clear separation of concerns between societal and agent-based features in the implementation of multiagent systems. From a theoretical perspective, this separation of concerns is also at the core of recent proposals towards a *societal* programming language. Building on the operational model of one of these proposals, this paper addresses the practical issue of implementing a web-based virtual machine for that language. The resulting framework is intended to be used in a wide range of applications, all of them related to the implementation of social processes (business processes, social networks, etc.).

1 Introduction

Different theoretical and practical insights into the field of computational organisations and electronic institutions [1,2,3,4] has led to a clear separation of concerns between societal and agent-based features in the implementation of multiagent systems. For instance, the institutional platform AMELI [5] makes a precise distinction between programming the e-institution (using the language of the ISLANDER tool) and programming the agents which participate in the e-institution (e.g. using the AgentBuilder tool). From a theoretical perspective, this separation of concerns is also at the core of recent proposals towards a *societal* programming language [3], which complements the myriads of *agent* programming languages that can be found in the literature (e.g. Jason [6], 3APL [7], etc.). The former kind of languages are aimed at programming socially-enable middlewares, whereas the later are aimed at programming agentified software components.

The design of a language for programming computational societies involves two major tasks: specifying the abstract social middleware – i.e. the abstract machine to be programmed, and specifying its type system. In [3], some preliminary steps towards the first goal are taken. Particularly, the proposed operational model of social interactions precisely states the structure and dynamics

[⋆] Research sponsored by the Spanish Ministry of Science and Education (MEC), project TIN2006-15455-C03-03, and the Regional Government of Madrid and University Rey Juan Carlos, project URJC-CM-2006-CET-0300.

of computational societies. Put in another way, it provides the structure and programmable behaviour of an abstract social middleware. Building on this theoretical results, this paper addresses the practical issue of implementing the language. The chosen middleware technology for implementing its virtual machine, i.e. the virtual middleware infrastructure, is the World Wide Web. This paper purports to present the architecture of a web-based social middleware infrastructure, namely the major design choices on its structure and dynamics. In order to attain this goal, the REST architectural style [8] and guidelines will be exploited.

The rest of the paper is structured as follows. Section 2 reviews from a middleware perspective the major concepts on the structure and dynamics of computational societies presented in [3]. Then, sections 3 and 4 address the architectural decisions on the structure and dynamics of a web-based social middleware infrastructure. The last section briefly summarises the major results and discusses current and future lines of work.

2 Computational Societies as Social Middleware Infrastructures

This section briefly reviews the operational model of computational societies put forward in [3] and introduces the example that will be used throughout the paper. Moreover, this operational model shall be interpreted as the abstract (i.e. technology-neutral) specification of a *social middleware* infrastructure. From this perspective, the major function of a computational society is to mediate the interactions among heterogeneous, distributed software components. In the next sub-sections, the kinds of *roles* played by software components attached to the social middleware as well as the primitive *interaction mechanisms* enabled by the middleware infrastructure will be considered. Moreover, the different types of external actions performed by software components over the middleware will be summarised. The last sub-sections introduce the abstract identifiers of middleware entities and an example within the university domain.

Roles. Software components interacting through an object-oriented middleware are published as *objects*; if the web is considered as the middleware infrastructure, software components play the role of *resources*; in a publish/subscribe infrastructure, components are attached to the middleware as *producers* and/or *consumers*; and so on. In regard with this feature, two kinds of roles are supported by a social middleware: *agents* and *resources*. On the one hand, resources represent those non-autonomous software components which store information and/or provide different computational services to the society. On the other hand, agents represent those autonomous software components which *purport* to achieve some goal within the society. In order to attain that goal, agents are able to perform different kinds of *social actions*, namely to say things to other agents (i.e. to perform communicative actions) and manipulate the environmental resources. The whole activity of some agent may be structured in a role-playing hierarchy of further agents (e.g. if its purpose is too complex).

Social Interactions. The interaction space of an object-oriented middleware is made up of *remote method calls*; the interactions through the web are handled in terms of *HTTP requests*; in the case of a publish/subscribe infrastructure, *event channels* are the primitive interaction mechanism. Concerning a social middleware infrastructure, its interaction space is hierarchically structured in terms of a tree of nested *social interactions*. In this way, the computational society itself is represented by the root, or top-level interaction. Social interactions provide the context within which agents and resources are deployed. Thus, a social interaction features a set of member agents, a set of environmental resources and a set of sub-interactions.

External Actions. External actions represent the interface between the abstract middleware and the external software components. Thus, a software component attached as an agent to the social middleware directs the behaviour of its agent through different kinds of external actions. For instance, the component may *play*/*suspend* its agent, thereby making public that the component is logged in/off the computational society. When the software component is logged in, it may *attempt* its agent to perform different kinds of *social actions*, e.g. *setting up* a new sub-interaction within a given context.

The social middleware deals with attempts in a three-stage process: firstly, it is checked whether the agent is empowered to do the specified social action (if it is not, the attempt is simply ignored); secondly, it is checked whether the agent is permitted to do the specified action under the current circumstances (if it is not, an event signalling the forbidden attempt is generated); last, if the agent is both empowered and permitted, the action is executed and the corresponding *events* signalling the updates in the social state are generated. These events may be notified to different agents according to their monitoring rules. In turn, these events may be pulled out by software components through the external action *observe*, which allows components to inspect the state of any social middleware entity[1].

Social Actions. The *set up* social action, mentioned above, is a communicative action (particularly, a declarative speech act) which is part of a predefined catalogue of standard social actions. This catalogue includes other actions, e.g., to prematurely finish a given sub-interaction (*close*); playing a new agent role within a given interaction context (*join*); and abandoning some played role (*leave*). Any kind of social action is *targeted* at some interaction whose state is intended to be modified (e.g. the target of a join action is the interaction context to which the performer intends to join). The protocol of the target interaction determines which agents are empowered and permitted to do that action.

Abstract Identifiers. Social middleware entities (agents, interactions, resources, etc.) have a unique abstract identifier. The abstract identifier of the

[1] The set of external actions mentioned above (*play*, *suspend*, *attempt* and *observe*) is complemented with other actions such as *enter* and *exit*, which deal with the registration of components to the middleware as software agents.

top-level interaction simply consists of a given *name*. Any other entity which is deployed within some interaction context is identified by a local *name* which identifies the entity within its interaction context, plus the identifier of its interaction context. Interaction identifiers are conventionally represented as a dot-separated sequence of local names, $n.n_c. \ldots .n_t$, which starts with its local name n and is followed by the context identifier $n_c. \ldots .n_t$; the sequence ends with the name n_t of the top-level interaction. Agent, resource, event and action identifiers are similarly represented. The only difference is that the name of the entity is separated from the context's identifier using the *at* sign ("@").

Example. Let's consider a social middleware to support the different social processes around the management of university *courses*. A given course, e.g. on data structures, is represented by a particular social interaction. On the one hand, this interaction is actually a complex one, made up of lower-level interactions. For instance, within the scope of a course agents will participate in *programming assignment groups*, *examinations*, *lectures*, and so on. On the other hand, courses are run within the scope of a particular *degree* (e.g. computer science), a higher-level interaction. Traversing upwards from a degree to its ancestors, we find its *school*, and finally the *university* (the top-level interaction). Besides schools, *departments* are also sub-interactions of the university. Taking into account the above structure of the interaction space, the identifier *ds.cs.si.urjc* stands for a course on data structures (*ds*) taught as part of the computer science degree (*cs*), managed by the school of informatics (*si*) at the University Rey Juan Carlos (*urjc*).

The agents within this computational society directly correspond to the different roles played by human users[2]. Thus, a student is represented by a role-playing hierarchy whose root is the *student* agent deployed within the degree; this student agent plays different student agent roles within the courses in which it has enrolled; in turn, students of courses may play corresponding roles within the programming assignment groups set up within their courses. Other agent roles, deployed within departments, include *associate professors* and *PhD candidates*, which play the roles of *teachers* and *teaching assistants* within courses, respectively. Concerning resources, we may consider different kinds of informational resources such as *programs* and *test cases*, generated by students within the context of working groups.

In the scenario that will be considered in the next sections, the agent *john@ds.cs.si.urjc* is a student of the course on data structures within the University Rey Juan Carlos. In order to pass the course (the purpose of course students), students have to pass several assignments in collaboration with another student. When the first assignment is published, *john*'s colleague sets up the working group *wg1.ds.cs.si.urjc*, which specifies *john* as an allowed partner. Then, an event representing this change is published and notified to *john*. When *john*'s human user observes these events, it attempts its agent *john* to join the assignment group. Then, since course's students are empowered to join working

[2] In this particular application, software components running the software agents are simply user interfaces, e.g. web browsers.

groups and *john* has been explicitly given permission, the action is executed by the middleware and a new agent *john@wg1.ds.cs.si.urjc*, played by *john*, is created within the assignment group.

3 Structure of a Web-Based Social Middleware Infrastructure

This section addresses the major design decisions concerning the *structure* of a web-based social middleware infrastructure, in accordance with the abstract specification introduced in the last section. The use of the web as the underlying distributed technology involves two major structural design problems:

- Firstly, computational societies must be published as web resources. These resources will represent the *entry* points to the social middleware for external software components.
- Secondly, different policies may be considered for the distribution of the interaction space through the network of web servers. Communication among socially-enabled web servers will rely on the previous *entry* points as well.

3.1 Publishing Social Entities as Web-Resources

There are several alternatives in order to expose a computational society through a web server. On the one hand, we may simply publish a single resource representing the whole computational society maintained by the web server. On the other, we may follow a fine-grained strategy and publish every major kind of social entity as a web resource. In order to leverage the HTTP protocol [9] to its full potential we follow the second approach. Thus, social interactions, agents, resources, actions and events are published as web resources. This allows, for instance, to implement the attempts of software components as HTTP POST requests over the agent resource, as described in the next section.

The URLs assigned to social entities follow general patterns which are designed after the structure of their corresponding abstract identifiers. Being exclusively based upon the data hold by abstract identifiers, the URLs resulting from these patterns are not expected to change very likely. Moreover, the URLs borrow the hierarchical and meaningfulness features of abstract identifiers as well. Table 1 shows the URL patterns assigned to the different kinds of social entities. Columns *Host:Port* and *Path* represent the corresponding parts of the URL. For every social entity, the *host* and *port* section of its URL represent the web server which manages that social entity. As will be described in the next subsection, with the possible exception of interactions, every entity is managed by the web server to which its interaction context belongs. The two rows of table 1 represent the two possible URL patterns:

- The first one is used for those interactions that are managed by a server different from its context's server. The URL for this kind of interactions is constructed by adding the name of the interaction to the root URL ('/')

Table 1. Generic Patterns of URLs

Entity	Name	Context	Host:Port	Path
Interaction	*name*	null	server.com:port	/*name*
Interaction, Event, Resource, Agent, Action	*name*	*interaction*	server.com:port	/path/to/*interaction*/*name*

of the server (note that the top-level interaction, whose context is empty, is a special case of this pattern).

– The second pattern applies to the entities that are published in the web server of its context interaction (i.e. events, actions, resources, agents and sub-interactions). The URLs of these entities are formed by appending the name of the entity to the URL of its interaction context (separated by '/').

3.2 Distributing the Interaction Space through Web Servers

We may consider two alternative stances on the distribution of the interaction space. The first one consists of ignoring this possibility so that the whole computational society is published through a single server. This means that this server processes all the HTTP requests over every social entity. In our scenario, this alternative forces a single host to process all the HTTP requests over the whole population of agents (students, teachers, etc.) and resources (assignments, solutions, plans of studies, etc.) of the university, as well as over its whole catalogue of social processes (courses, departments, schools, etc.). This alternative is only valid for applications with low demands for scalability, where the population of agents and resources as well as their interactions are kept under strict limits.

The second alternative, advocated by this paper, consists of allowing the distribution of the interaction space through multiple servers. The use of URL-addressable resources allows the distribution of the computational society over the Web, thereby exploiting its potential for scalability. The only constraint imposed on the distribution is that every social entity, but interactions, must be deployed within the web server which manages its interaction context. Sub-interactions may be deployed within the web server which manages its interaction context, but this is not mandatory. On the contrary, the URLs of agents, resources, actions and events always share the *host:port* part with the URL of their interaction context. Without any further restriction, and with the intention of guaranteeing the maximum deployment flexibility, every socially-enabled web server is allowed to manage a forest of interaction trees. Each of them may belong or not to the same computational society.

For instance, figure 1 shows a possible deployment of the interaction space corresponding to the scenario described in section 2. Particularly, it depicts the distribution of the major interactions throughout four hosts, each of them running a single web server. The first host, *www.univhost.com*, manages the top-level interaction *urjc*, representing the university itself. According to the patterns described in section 3.1, its URL is *http://www.univhost.com/urjc*. The second host, *www.schoolhost.com*, manages that part of the interaction space which is under the primary responsibility of the school of informatics, namely the social interaction representing the school itself *si.urjc* and its different degrees. Besides the computer science degree *cs.si.urjc*, shown in the figure, other degrees such as software and computer engineering may be published through this host as well. The URLs of the school of informatics and the computer science degree are *http://www.schoolhost.com/si* and *http://www.schoolhost.com/si/cs*, respectively. The third host, *www.depthost.com*, is associated to the computer science department of the university, *csd.urjc*, published under the root URL of the host *http://www.depthost.com/csd*. In accordance with the statutes of the university, departments are in charge of the management of courses on the different subjects which are assigned to them. Thus, the course on data structures *ds.cs.si.urjc* is published through the computer science department under the URL *http://www.depthost.com/ds*. In this case, the web server of the department host manages two sub-interaction trees. The last host, *www.studenthost.com*, manages the working group set up by one of the students enrolled in the data structure course. The web server of the student's host may manage different sub-interaction trees from other computational societies as well (e.g. a discussion forum set up by the student within a social network).

4 Dynamics of a Web-Based Social Middleware Infrastructure

The dynamics of a computational society is primarily influenced by the external actions which software components execute over the social middleware which manages that society. Since the social middleware is implemented as a network of socially-enabled web servers, external actions are implemented as different kinds of HTTP requests. Moreover, the activity carried out by the middleware in order to process the different external actions heavily relies in the HTTP protocol as well. This is a direct consequence of the distribution of the computational society across the network of web servers. Therefore, HTTP requests may represent either an external action or some internal action executed by the middleware as part of the external action processing. Both kinds of HTTP request are handled through a pool of *conceptual* execution threads. These threads have a one-to-one correspondence to the different kinds of social entities. Thus, the agent execution thread processes every HTTP request whose target URL denotes an agent resource. Similarly, the interaction, resource, action and event execution threads manage the HTTP requests addressed to the corresponding kinds of social entities.

The remainder of this section proceeds to describe the implementation of the external actions mentioned in section 2. Particularly, it will be described both the way in which a given external action is represented as an HTTP request and the roles played by the different conceptual threads involved in its processing. The external actions *play* and *suspend*, related with *login* features, are taken into account first. Next, the mapping and internal processing of *observation* actions is

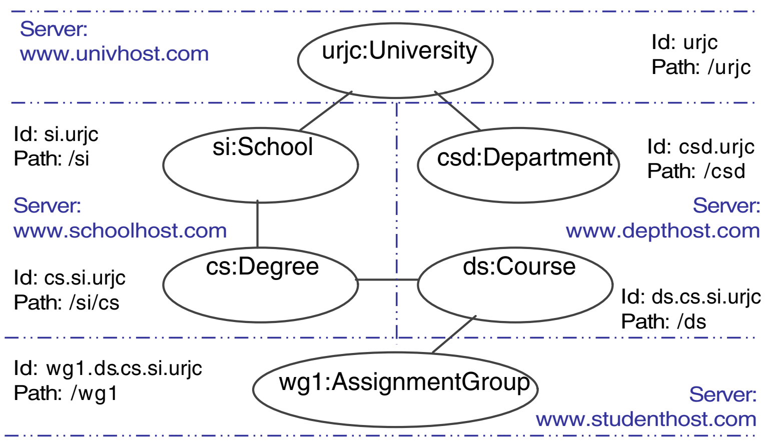


Fig. 1. Distribution of the example's interaction space

considered. Finally, we present the *attempt* processing cycle. Before delving into the different external actions, however, the major HTTP methods and response status are briefly summarised.

Review of HTTP. HTTP is a client-server protocol: a client sends a request message to a server, which does some processing and returns afterwards a response message containing a status code and the result of the request (or information about the status code). The format of a request message consist of a request line, zero or more header lines, and an optional message body. Both the standard semantics of status codes and HTTP headers are explained in [9]. A request line has three parts, separated by spaces: a method name, the local path of the requested resource (*Request-URI*), and the version of HTTP being used. A typical request is:

```
METHOD /path/to/resource HTTP/1.1
Header: value
...
Message-Body
```

HTTP Methods. The semantics of the request rely on the chosen HTTP method. We restrict our review to the four basic HTTP methods: GET, POST, PUT and DELETE.

– GET: This method is intended to obtain a representation of the resource identified by the Request-URI. It can be parameterized in order to constrain or restrict the desired representation.
– POST: This method is used both to create new resources and to append data to an existing resource. If the method is used to create new resources the body of the request will contain an entity. This entity must be created

by the resource identified by the Request-URI and the decision about the URL of the new entity is left to the server. On the contrary, if it is used to append data, the body of the request will represent data that must be added or processed by the Request-URI.
– PUT: The PUT method is used for creating a new resource (or updating the state of an existing one) under the supplied Request-URI. The message body of the request encodes the entity that will be published. If an entity already exists on the Request-URI, then the message body encapsulates an update of the entity (either full, affecting to the totality of their attributes, or partial).
– DELETE: This method unbinds a resource from the specified Request-URI. Note that this method does not imply the deletion of the actual data held by the resource or the software component behind it.

Headers. HTTP defines 47 headers which add optional meta-information about the Message-Body or, if no body is present, about the resource identified by the request. The most relevant header from the point of view of this paper are the following: *Authorization*, *Host*, *Location*, *Referer* and *WWW-Authenticate*.

Status Codes. Response messages to HTTP requests consists of a Status-Code element, some headers and a message body. The Status-Code is a 3-digit integer code which represents the result of the attempt made by the server to understand and satisfy the request. The first digit of the Status-Code defines the class of response (1xx informational, 2xx success, 3xx redirection, 4xx client error, 5xx server error). The body of response messages may give a short textual description of the Status-Code. Some of the codes we use in this paper are: *200 ("OK"), 201 ("Created"), 202 ("Accepted"), 204 ("No Content"), 400 ("Bad Request"),401 ("Unauthorized"), 403 ("Forbidden").*

Play Processing. The external action *play* is used by a software component to initiate a logging session with a given agent, thereby obtaining the corresponding credentials to manipulate it. To log in is a mandatory requirement for performing some external actions such as *attempts*. Other actions, however, can be executed by non-logged components (e.g. *enter*, *observe*, the *play* action itself, etc.).

A digest access authentication scheme is proposed for dealing with credentials [10]. This scheme assumes that component credentials consist of the top-level agent name, a password defined when the agent was registered, a unique value shared between server and client and the MD5 algorithm. Once the component has got credentials, it may include them in every subsequent request using the *Authorization* header. The *play* external action is implemented as a GET request about the credentials of the agent, targeted over the URL of the agent.

```
GET /path/to/agent/credentials HTTP/1.1
Host: www.server.com
```

This action is processed by the agent execution thread. If some component has already initiated a session with the agent, the request is ignored and a response with status code 400 is sent back. Otherwise, a response with status code 401 is returned. In this response, a WWW-Authenticate header includes the protected realm (the top-level agent), a unique value named *nonce* and a digest algorithm.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest realm="top-agent's name",
                  nonce="84e0a095cfd25153b2e4014ea87a0980",
                  algorithm=MD5
```

In subsequent requests, the component composes a valid credential applying the MD5 algorithm to a combination of the top-agent's name, the nonce value and the password, among other parameters. The server does the same computation and if it yields the same credentials, it can be sure that the component is in possession of the correct password.

Suspend Processing. Components can finalise its session with some agent with the *suspend* external action. This action deletes the nonce value associated with the component. Afterwards, the credentials of the server and the client won't match and the component's authorization credentials will be invalid. This external action is translated as a PUT action over the agent's credentials URL with a null message body.

```
PUT /path/to/agent/credentials HTTP/1.1
Host: www.server.com
```

This request is processed by the agent execution thread. This thread deletes the current nonce value, invalidating the following requests of the component. The response's status code is 204, because the server executes the action but declines to send back any representation. At a later time, a component may *play* again its agent, thus renewing its agent credentials.

Observe Processing. Components can get representations of the different published entities (interactions, agents, resources, etc.) through the *observe* external action. This action returns a representation of the requested entity, restricted according to the visibility rules of the society. This external action is translated as a GET query over the URL of the entity. This query can be parameterized to select some parts of the resource instead of its full representation.

```
GET /path/to/entity HTTP/1.1
Accept:
Host: www.server.com
```

This request is processed by the execution thread corresponding to the kind of entity to be inspected. This thread checks the protocol for visibility restrictions and returns a response with status code 200. The message body of the response contains the representation of the resource (maybe restricted according to its visibility permissions). This method can be executed by any component. If the component has initiated a session with an agent, the obtained representation will be tailored to the visibility permissions corresponding to agents of that type; otherwise, if the component hasn't got any agent credentials the obtained representation is the one associated by the protocol to agents of any type.

For instance, figure 2 shows a sequence diagram depicting the activity of the middleware in response to the scenario introduced in section 2. The roles displayed in the sequence diagram represent the different published resources; their lifelines describe the activity of the threads that manage those kinds of entities. The sequence diagram assumes that the student John has previously initiated a session as a student of the course on data structures (*ds*) with the web server `http://www.depthost.com` of the department of computer science, using a web `Browser`. The first message shows John *observing* the event queue of its agent *http://www.depthost.com/ds/john*, i.e. John's client producing the HTTP request *GET /ds/john?show=events* (message 1)[3]. Then, the response includes a representation of the event queue including the URLs of the events received by the agent (message 2). One of them refers to the new assignment group `http://www.studenthost.com/wg1` set up by its colleague. The remaining messages pertain to the processing of the attempt made by John to make its student join this assignment group.

Attempt Processing. The external action *attempt* aims at *adding* a new pending action to the specified performer agent. According to this specification, this kind of external action is translated as a POST request over the agent URL. The attempt data (the action as well as other attributes) is included in the message body:

```
POST /path/to/agent HTTP/1.1
Host: www.server.com

<attempt>
 <action>...</action>
 ...
</attempt>
```

Message 3 of figure 2 represents an instance of the previous HTTP pattern. Particularly, it refers to the HTTP request corresponding to the attempt of John to make its student agent join the assignment group set up by its colleague.

An *attempt* HTTP request may refer to an action which is not targeted at some interaction managed by the same web server of the performer agent. Since the protocol of the target interaction must be consulted to check the empowerments of the agent, the agent execution thread processes an *attempt* HTTP request by requesting the target interaction to create it. Particularly, the request is actually

[3] Commonly, the request will actually be generated by the user interface components of the web browser, e.g. Java script code.

issued through a POST method over the target interaction URL[4]. The request includes the *referrer* header that indicates the performer agent of the action. The message body of this request is the action; the name of the action is set by the agent execution thread based on the reserved word "act", the performer's name and an incremental counter. Message 4 of figure 2 shows an instance of the following request pattern:

```
POST /path/to/target/interaction HTTP/1.1
Host: www.server.com
Referer: http://www.maybeotherserver.com/path/to/performer

<action name="act_performer_1">...</action>
```

If the performer is not empowered to do the action then a request with status code 400 is returned. Otherwise, the interaction execution thread creates the action for the specified performer agent – in the *referrer*'s server. This is encoded using a PUT action over the interaction context's URL. The content of the message body is the action as shown in the following scheme:

```
PUT /path/to/referrer/interaction/act_performer_1 HTTP/1.1
Host: www.server.com

<action name="act_performer_1">...</action>
```

For instance, message 5 of figure 2 shows the creation of a new action resource in the server of the computer science department, in accordance with the protocol's empowerment rules of assignment group interactions. Then, the action execution thread of the computing department server processes this request, creates the action, and sent back a response with a status code 201 with the action URL in the *Location* header (message 6). This response is forwarded by the interaction execution thread of the student server to the agent execution thread of the computing department server (message 7), which finishes the processing of message 4. Then, the student's execution thread sends a response back to the browser with a status code 202 and the corresponding *Location* header (message 8), which finishes the processing of the attempt HTTP request (message 3). This status code indicates that the action has been just accepted for execution. The *Location* header refers to the action resource as the monitor to check the processing state.

The action execution thread is responsible for the execution of the social action as soon as it is created. The way in which the action is executed depends on its semantics. Nevertheless, an HTTP request will be involved which must contain a *Referer* header with the URL of the action performer. For instance, the execution of a *join* action involves the creation of a new agent resource in

[4] Thus, the target interaction is acting here as an action *factory*.

the target interaction. Therefore, the corresponding HTTP request will be a PUT request which attempts to create a new agent resource. The message body contains the description of the new agent instance:

```
PUT /path/to/new/agent HTTP/1.1
Host: www.server.com
Referer: http://www.maybeotherserver.com/path/to/performer

<agent>...</agent>
```

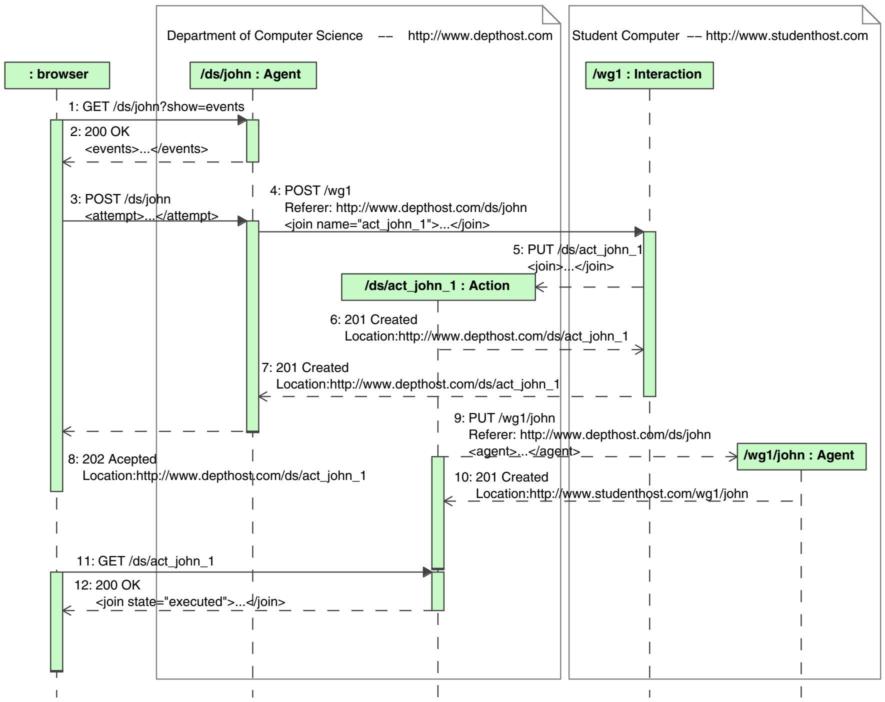

Department of Computer Science -- http://www.depthost.com
Student Computer -- http://www.studenthost.com
: browser
/ds/john : Agent
/wg1 : Interaction
1: GET /ds/john?show=events
2: 200 OK
<events>...</events>
3: POST /ds/john
<attempt>...</attempt>
4: POST /wg1
Referer: http://www.depthost.com/ds/john
<join name="act_john_1">...</join>
5: PUT /ds/act_john_1
<join>...</join>
/ds/act_john_1 : Action
6: 201 Created
Location:http://www.depthost.com/ds/act_john_1
7: 201 Created
Location:http://www.depthost.com/ds/act_john_1
9: PUT /wg1/john
Referer: http://www.depthost.com/ds/john
<agent>...</agent>
/wg1/john : Agent
8: 202 Acepted
Location:http://www.depthost.com/ds/act_john_1
10: 201 Created
Location:http://www.studenthost.com/wg1/john
11: GET /ds/act_john_1
12: 200 OK
<join state="executed">...</join>

Fig. 2. Join to an Assignment Group

For instance, message 9 of figure 2 shows the PUT request issued by the action execution thread of the computing department server. This request aims at creating the agent `/wg1/john` within the assignment group of the student's server. The request is processed by the agent execution thread[5], which checks if the referer agent (i.e. the performer of the action) has permissions to execute the action.

[5] Requests corresponding to other actions, such as *leave*, *set up*, *close*, etc. would be processed by other threads. For instance, *set up* and *close* involves PUT and DELETE requests over web interaction resources. Therefore, these methods would be processed by the interaction execution thread.

If it is not permitted then a response with a status code 403 is returned. Otherwise, the action is executed and a suitable response is generated. Message 10 of figure 2 shows the successful creation of the agent, which means that John's agent was permitted to join the assignment group. Then, the action execution thread changes the state of the action accordingly. The scenario is finished when John (actually, the web browser) *observes* the state of the action execution (using the URL monitor sent back in message 8), through messages 11 and 12.

5 Conclusion

This paper has put forward some of the major architectural decisions in the development of a web-based middleware infrastructure for the implementation of multiagent societies. Firstly, a computational society is published in the web through the refinement of web resources into three major sub-kinds: *web agents*, *web (institutional) resources* and *web interactions* (i.e. processes). Social actions and events are also published as web resources. Secondly, to account for *scalability*, *privacy*, and *flexibility of deployment* requirements, computational societies are allowed to be distributed across different socially-enabled web servers, each of them managing a forest of subinteraction trees. Last, in order to exploit the HTTP protocol in its full potential, the major HTTP methods (GET, PUT, POST and DELETE) are used to implement the external actions performed by software components towards the computational society, and the internal processing of the socially-enabled web servers.

One of the distinctive features of the proposed agent-based middleware infrastructure is the use of the web as the underlying middleware technology. On the contrary, other approaches face the web as a complementary – not fundamental – distributed infrastructure. In these contexts, the web appears in the issue of interoperability between agents and web service components (e.g. [11]). In our view, using the web as the underlying distributed infrastructure presents two major advantages: firstly, software components acting as agents within the society can be entirely *de-coupled* from the middleware server, which fits well with the autonomy requirement of agents; secondly, the web is one of the largest deployed distributed infrastructures, so that the network of social servers have not to be built from scratch.

Current work focuses on the implementation of the proposed architecture using the Restlet framework [12]. The resulting framework is intended to be used in a wide range of applications, all of them related to the implementation of social processes: business processes, e-government, e-democracy, etc. Particularly, we intend to demonstrate the feasibility and potential of the social stance of multiagent technologies on distributed computing, as well as the web-based approach to their implementation proposed in this paper, in the programming of social networks. Social networks like Facebook, Myspace, LastFM, etc., provides its users with different interaction mechanisms (chats, discussion groups, etc.). The modification and extension of these social networks essentially involves programming new social interactions. A web-based, social-oriented approach to the implementation of social networks would make this task much easier.

References

1. Ferber, J., Gutknecht, O., Michel, F.: From agents to organizations: An organizational view of multi-agent systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
2. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. ACM Transactions on Software Engineering and Methodology 12(3), 317–370 (2003)
3. Serrano, J.M., Saugar, S.: Operational semantics of multiagent interactions. In: Proceedings of the Sixth Intl. Joint Conf. on Autonomous Agents and Multiagent Systems, Honolulu, Hawai'i, 14-18 May 2007, pp. 884–891. ACM Press, New York (2007)
4. Esteva, M., Rodriguez, J.A., Sierra, C., Garcia, P., Arcos, J.L.: On the formal specifications of electronic institutions. In: Sierra, C., Dignum, F.P.M. (eds.) Agent-mediated Electronic Commerce (The European AgentLink Perspective). LNCS (LNAI), vol. 1991, pp. 126–147. Springer, Heidelberg (2001)
5. Esteva, M., Rosell, B., Rodríguez-Aguilar, J.A., Arcos, J.L.: AMELI: An agent-based middleware for electronic institutions. In: Proc. 3rd. Int. Joint Conf. on Autonomous Agents and Multiagent Systems, vol. 1, pp. 236–243 (2004)
6. Bordini, R.H., Hübner, J.F., Vieira, R.: Jason and the golden fleece of agent-oriented programming. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Platforms and Applications, Springer, Heidelberg (2005)
7. Hindriks, K.V., Boer, F.S.D., der Hoek, W.V., Meyer, J.J.C.: Agent programming in 3APL. Autonomous Agents and Multi-Agent Systems 2(4), 357–401 (1999)
8. Fielding, R.T., Taylor, R.N.: Principled design of the modern web architecture. ACM Trans. Inter. Tech. 2(2), 115–150 (2002)
9. Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext transfer protocol – HTTP 1.1 (1999)
10. Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A., Stewart, L.: Http authentication: Basic and digest access authentication (1999)
11. JADE: Jade web services integration gateway (2007), http://jade.cselt.it
12. Consulting, N.: Restlet - lightweight rest framework for java (2007), http://www.restlet.org

Using the Wizard of Oz Method to Train Persuasive Agents

Maiko Kawasoe, Tatsuya Narita, and Yasuhiko Kitamura

School of Science and Technology, Kwansei Gakuin University
2-1 Gakuen, Sanda-shi, Hyogo 669-1337, Japan
ykitamura@kwansei.ac.jp

Abstract. Persuasive conversational agents persuade users to change their attitudes or behaviors through conversation and are expected to be applied as virtual sales-clerks in e-shopping sites. Developing such an agent requires a conversation model that identifies the most appropriate responses to the user's inputs. To create such a model, we propose the approach of combining a learning agent with the Wizard of Oz method; in this approach, a person (called the Wizard) talks to the user pretending to be the agent. The agent learns from the conversations between the Wizard and the user and constructs its own conversation model. In this approach, the Wizard has to reply to most of the user's inputs at the beginning, but the burden gradually falls because the agent learns how to reply as the conversation model grows.

Every persuasive conversation has the goal of persuading the user and ends with success or failure. We introduce a goal-oriented conversation model that can represent the success probability of persuasion and a learning method to update the model depending on the success/failure of the persuasive conversation. We introduce a learning persuasive agent that implements the conversation model and the learning method and evaluate it in the situation wherein the agent persuades users to choose one type of digital camera over another. The agent could succeed in reducing the Wizard's inputs by 48%, and, more interestingly, succeeded in persuading 2 users without any help from the Wizard.

1 Introduction

Persuasive technology draws attention as a means to create interacting computing systems that can change people's attitudes and behaviors [1]. Conversational agents will play an important role in such systems. They can interact with users through conversation [2] and are expected to become virtual sales-clerks that persuade customers to Web shopping sites [3].

Developing a conversational agent requires a conversation model that represents how the agent responds to inputs from users. It is not easy to create a conversation model in which the agent interacts well with users and a large number of conversation rules must be created by experts. To reduce the burden, we integrate a learning agent and the Wizard of Oz method [4], in which a person called the Wizard talks with a user pretending to be the agent. The agent learns from the conversations between the Wizard and the users and constructs/refines a conversation model. At the beginning, the Wizard

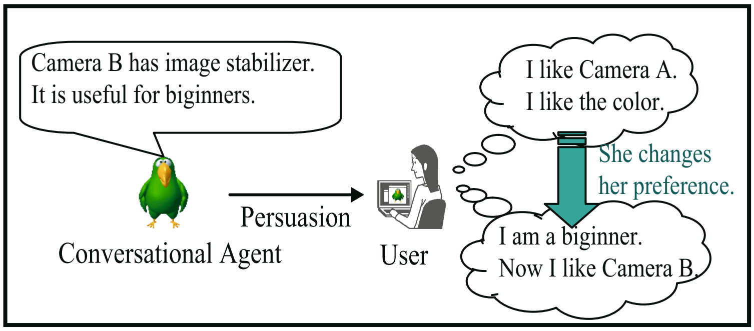


Fig. 1. Persuasion through conversation

has to input most of the replies, but gradually the agent learns to reply appropriately as the conversation model grows. When a reply made by the agent is not appropriate, the Wizard can correct it.

In this paper, we introduce a conversational agent that persuades users as shown in Fig. 1. The user initially prefers Camera A over Camera B, and the agent tries to persuade her to change her preference from A to B.

Every persuasive conversation has the goal of persuading a user and ends with success or failure. We introduce a goal-oriented conversation model that can represent the success probability of persuasion and a learning method to update the model depending on the success/failure of the persuasive conversation.

Section 2 of this paper addresses conversational agents and the Wizard of Oz method as the bases of persuasive conversational agents. In Section 3, we propose a goal-oriented conversation model and a learning method to update the model considering the success probability of persuasion. We then show a prototype system in Section 4 and evaluation results in Section 5. Finally, we conclude this paper with our future work in Section 6.

2 Persuasive Conversational Agents

2.1 Conversational Agents

Conversational agents interact with users though conversation to assist them in their information processing tasks such as information retrieval from the Web [3]. ALICE (Artificial Linguistic Internet Computer Entity) is representative of the conversational agents now available on the Web and is being used in a number of Web sites.[1]

The conversation model represents how an agent replies to inputs from users. There are two major approaches to constructing a conversation model. The first one is by describing scenarios or rules as is used in ALICE, PPP Persona [5,6], and so on. ALICE uses a language called AIML (Artificial Intelligence Markup Language), based on XML, to describe rules, each of which links a pattern, which represents an input from

[1] http://www.alicebot.org/

the user, to a template, which represents a reply from the agent. This approach forces us to write a large number of rules to make the agent reply fluently to various inputs from the user.

The second approach is to utilize a conversation corpus as is done in Command Talk [7]. In this approach, we need to establish a very large conversation corpus in advance to construct a conversation model. However, the agent cannot reply appropriately to an input if the input is not in the corpus.

2.2 Wizard of Oz Method

This paper takes the approach of integrating a learning agent and the Wizard of Oz method [8] as shown in Fig. 2. In the Wizard of Oz method, a person called the Wizard interacts with the user pretending to be the conversational agent. The Wizard can reply to input from the user when the agent cannot reply appropriately. The agent learns from the Wizard how to reply to an input by constructing a conversation model and can thereafter reply to the next instance of the same input. At the beginning, the Wizard has to reply to most of the inputs, but the burden of the Wizard falls because the agent learns to reply as the conversation model matures.

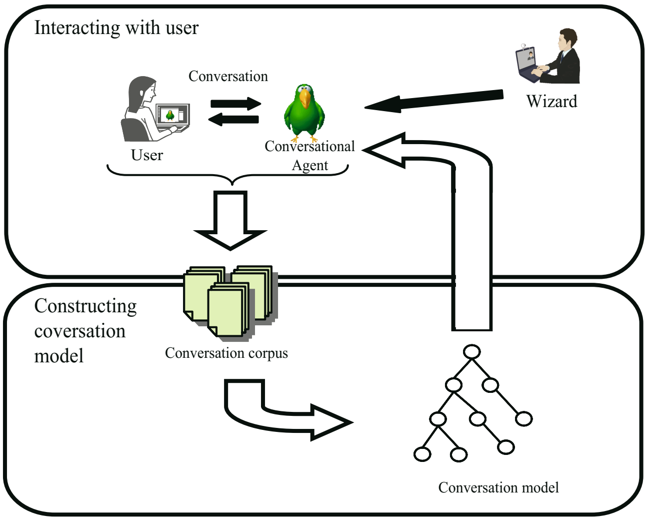


Fig. 2. Integrating a learning agent and the Wizard of Oz method

2.3 Persuasive Conversation

Persuasion is the action of changing people's attitudes and behaviors [1]. This paper considers the example of an agent that tries to persuade a user to change his/her preference from camera A to Camera B. If the user comes to prefer B, we define the persuasion as successful; otherwise, a failure.

Conventional conversational agents reply to an input from a user if the input matches a rule in the conversation model. When it matches multiple rules, one of them is selected. The selection process depends on the system and/or the applied domain. Persuasive agents, on the other hand, should select the rule that is more likely to lead to success. To this end, we propose a goal-oriented conversation model that considers the success probability of persuasion and a learning method to update the probability as derived from persuasive conversations between the Wizard and users.

3 Learning Persuasive Agents

To build persuasive agents that can learn, we need a goal-oriented conversation model and a learning method that can update the conversation model. Details of the model and the learning method are given below.

3.1 Goal-Oriented Conversation Model

The conversation model can be represented as a state transition tree where a statement is represented as a link to change a state from one to another as shown in Fig. 3. In this example, the agent tries to persuade a user to be a member of Kitamura laboratory. There are two types of states; user states, which represent the user talking and agent states, which represent the agent talking. They are interleaved on the conversation path. A conversation path represents the flow of conversation between the agent and one or more users and begins with the initial states and terminates with either success or failure. Each state is assigned a success probability score.

The agent decides how to respond to an input from the user following the conversation path held by the model. If the input matches a statement on a link to an agent state,

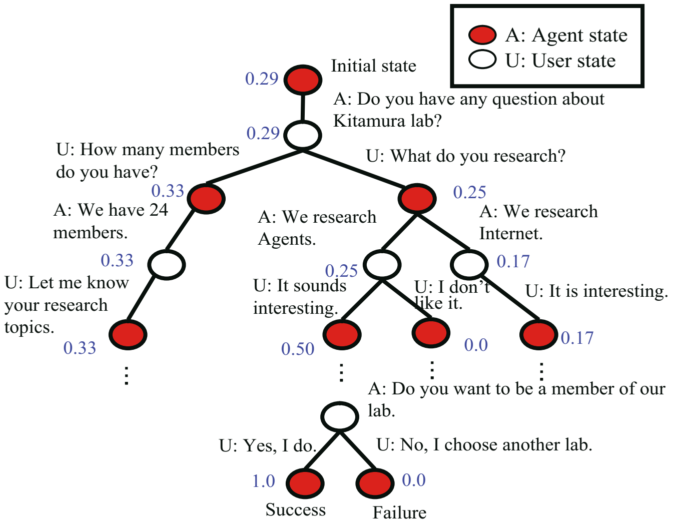


Fig. 3. Conversation model

it chooses a statement that links the agent state to the user state with greatest success probability.

For example in Fig. 3, the agent says "Do you have any question about Kitamura lab?" at the beginning. If the user asks "How many members do you have?" the agent replies "We have 24 members," following the stored conversation path. If the user asks "What do you research?" there are two reply candidates. The agent chooses the reply "We research Agents." because that link leads to a user state with higher success probability (0.25).

3.2 Updating Conversation Model

When an input from the user does not match any statement on the stored conversation path, the conversation path is branched and the success probability scores are updated depending on persuasion success/failure as shown in Fig. 4. If the persuasion succeeds (fails), 1.0 (0) is assigned to the terminal state. The success probability score of each state except terminal states in the conversation model is updated as below.

– Agent state s

$$Q(s) \leftarrow \max_{t \in succ(s)} Q(t)$$

– User state s

$$Q(s) \leftarrow \frac{1}{|succ(s)|} \sum_{t \in succ(s)} Q(t)$$

$succ(s)$ is a set of child states of s. At an agent state, the agent can choose what to say, so the success probability is set to be the maximum one among child user states. On the other hand, at a user node, the user chooses what to say, so the success probability is set to be the average one among child agent states. We here assume that the user takes a neutral attitude toward the agent. If we assume the user takes a negative attitude, the success probability should be the minimum one.

For example, when an agent says "We research Agents." using the conversation model shown in Fig. 3, if the user replies "What are Agents?" which is not contained in the model, a new conversation path is created by branching as shown in Fig. 4 following a persuasive conversation between the Wizard and the user. The persuasion succeeds, so 1.0 is attached to the terminal state of the branched path and each state on the conversation path is updated as mentioned above.

3.3 Reducing Redundancy in the Conversation Model

Because our conversation model is a tree, two conversation paths that virtually identical are treated as completely different if the first parts of the paths are different. Naive extension of the conversation model creates redundancy.

To reduce this redundancy, we transform the conversation model by using multiple phases as shown in Fig. 5. In this example, two conversation paths are different even though only the first parts are different as shown in Fig. 5 (a). We transform the model into one with two phases; greeting phase and persuasion phase, as shown in Fig. 5 (b) to reduce the redundancy.

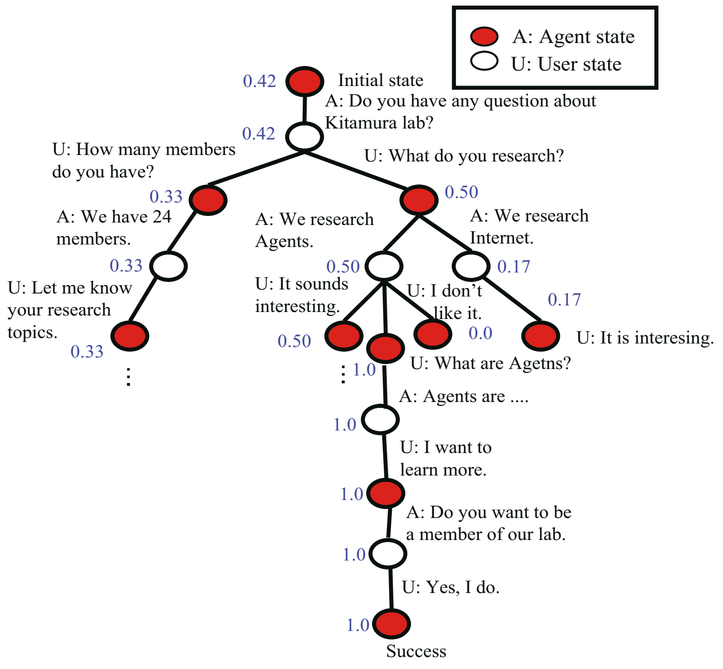


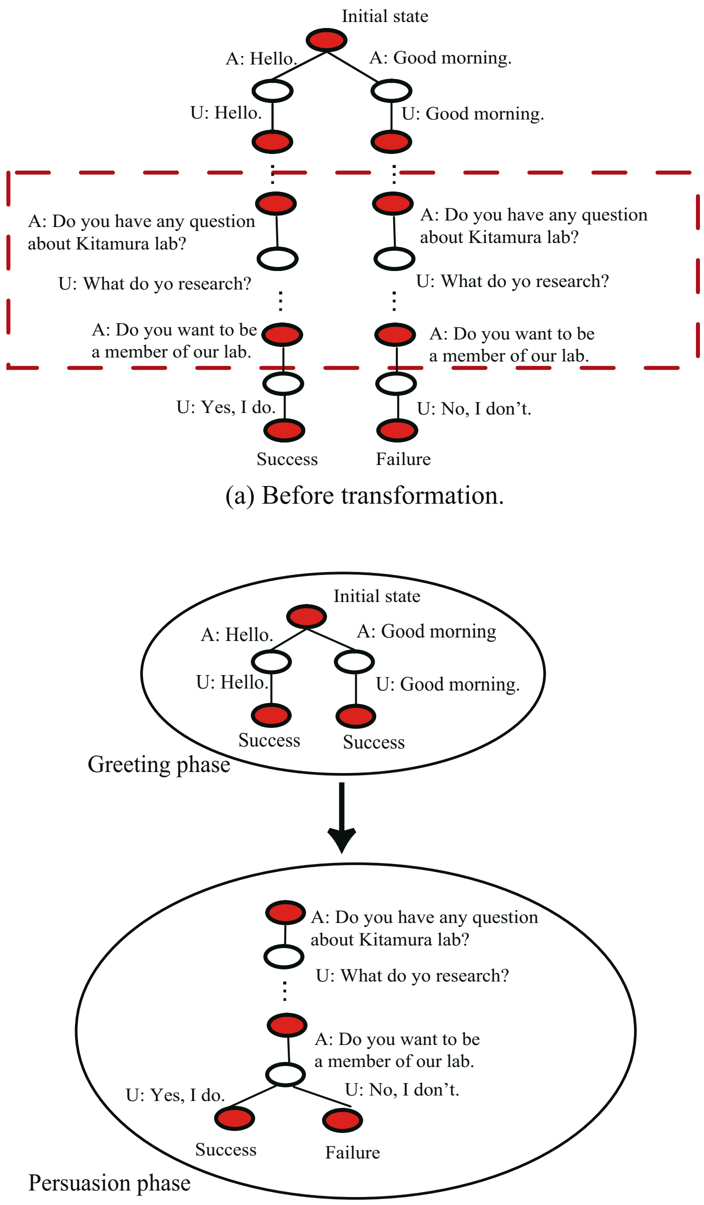
Fig. 4. Updated conversation model

4 Implementing a Persuasive Conversational Agent

We implemented a persuasive conversational agent as shown in Fig. 6 to chat with a user. Messages from the agent appear in the top panel and the user inputs messages to the agent in the message box at the bottom as shown in Fig. 7. When the agent receives a message from the user, it generates responses from the conversation model. There are two types of responses.

Context sensitive responses (CSR) are generated by the model by following a conversation path from the initial state. For example in Fig. 3, if the user inputs "What do you research?" in response to the message "Do you have any question about Kitamura lab?" from the agent in the initial state, the agent generates two context sensitive responses "We research Agents," and "We research Internet." Their success probabilities are 0.25 and 0.17, respectively.

Context free responses (CFR) are generated by the model by direct matching of the input from the user without following any conversation path. For example in Fig. 3, after the interchange of; "Do you have any question about Kitamura lab?", "How many members do you have?", and "We have 24 members," if the user asks "What do you research?" the agent generates two context free responses "We research Agents." and "We research Internet." In this case, because the responses do not follow any conversation path, success probabilities are not attached to the responses.



(a) Before transformation.

(b) After transformation.

Fig. 5. Transforming conversation model

In the early stages of learning, the agent often fails to respond to the user if it uses only CSRs because the conversation model is small. By utilizing CFRs, the agent can generate more candidates.

Messages from the user appear on the top panel of the Wizard chat client and responses generated by the agent appear as in a pull-down menu as shown in Fig. 8. The Wizard can choose the most appropriate one from among them. If the Wizard does not like any response, he/she can input a new message directly into the message box.

When a persuasion succeeds/fails or a phase terminates, the Wizard notifies the state to the system through the pull-down menu in the left-bottom and a textbox to specify the next phase as shown in Fig. 9. A button on the right-bottom is used to show the conversation model, an example of which is shown in Fig. 10. When a link is clicked, the corresponding message appears in the window.

5 Evaluation

We evaluated our persuasive conversational agent from two viewpoints; (1) the input cost of Wizard when utilizing responses created by the agent, and (2) the persuasiveness of the conversation model constructed through conversations between users and the Wizard.

We performed two experiments in which the persuasive conversational agent tried to guide users to choose one of two digital cameras using the following procedure.

1. Each participant read the specifications of two digital cameras, A and B, as shown in Table 1. Camera A has better features than camera B, but the price of A is more than that of B.
2. The participant chooses one as his/her favorite from the first impression.
3. The agent, with help from the Wizard, tries to persuade the participant to choose the other one. It first asks why he/she chose the one selected, and then it tries to refute the reasoning. It asks him/her for situations in which he/she would use the camera and his/her taste, and recommends the camera that he/she did not initially choose.

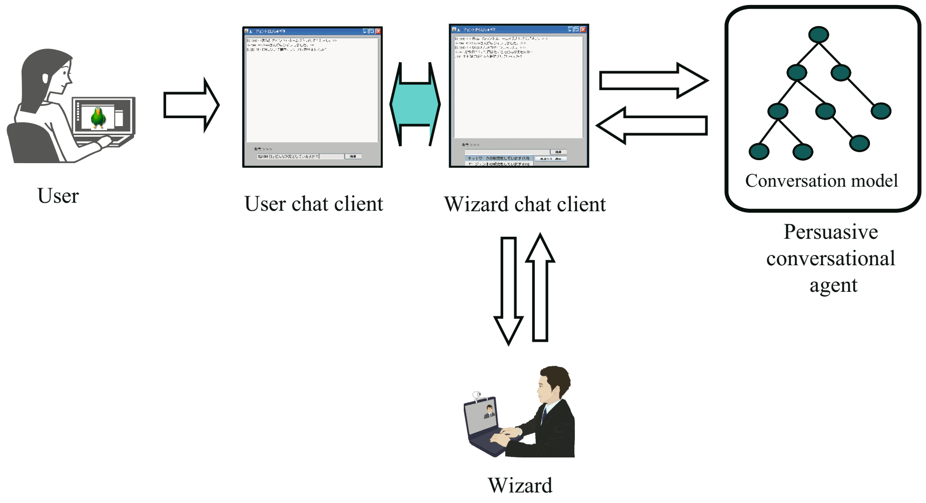
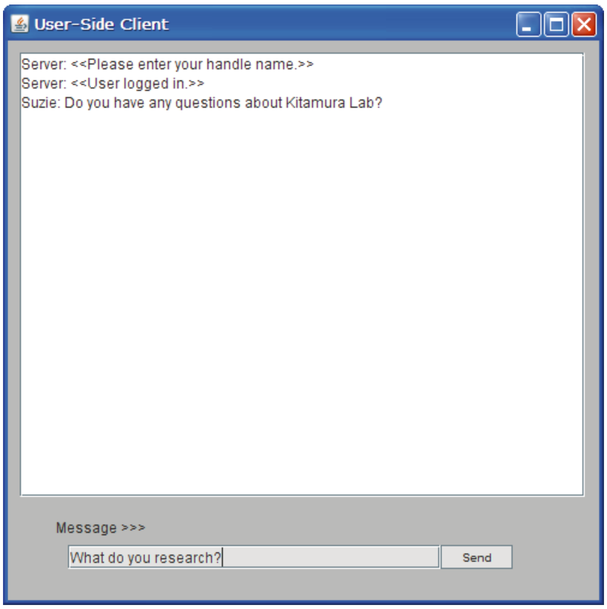


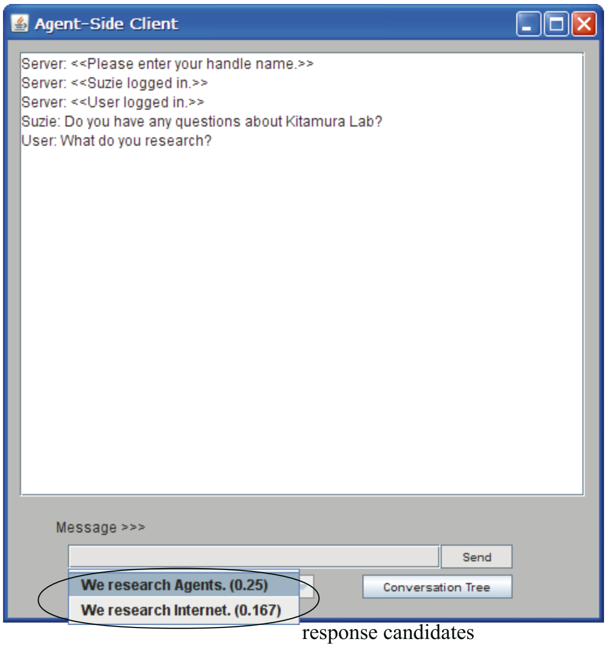
Fig. 6. System overview



Server: <<Please enter your handle name.>>
Server: <<User logged in.>>
Suzie: Do you have any questions about Kitamura Lab?

What do you research?

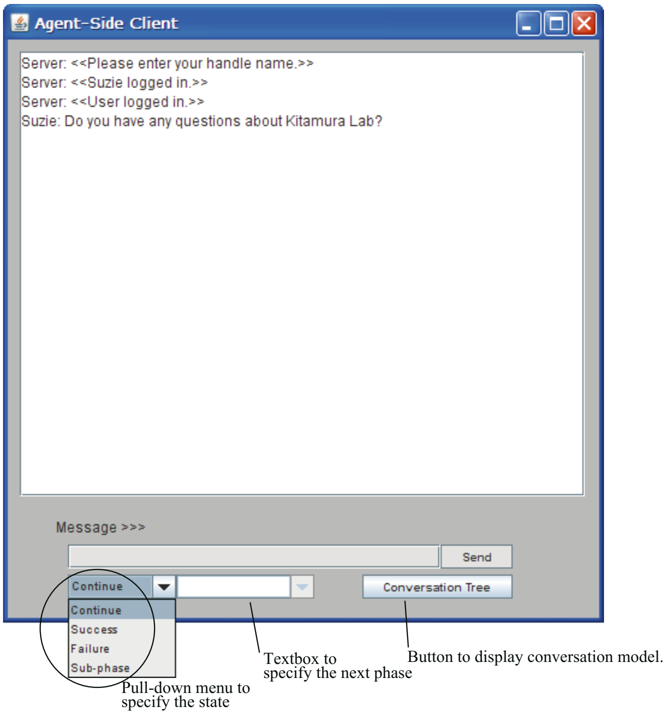
Fig. 7. User chat client



Server: <<Please enter your handle name.>>
Server: <<Suzie logged in.>>
Server: <<User logged in.>>
Suzie: Do you have any questions about Kitamura Lab?
User: What do you research?

We research Agents. (0.25)
We research Internet. (0.167)

Fig. 8. Wizard chat client: choosing a response


Agent-Side Client
Message >>>
Send
Continue
Conversation Tree
Continue
Success
Failure
Sub-phase
Textbox to specify the next phase
Button to display conversation model.
Pull-down menu to specify the state

Server: <<Please enter your handle name.>>
Server: <<Suzie logged in.>>
Server: <<User logged in.>>
Suzie: Do you have any questions about Kitamura Lab?

Fig. 9. Wizard chat client: specifying the state transition

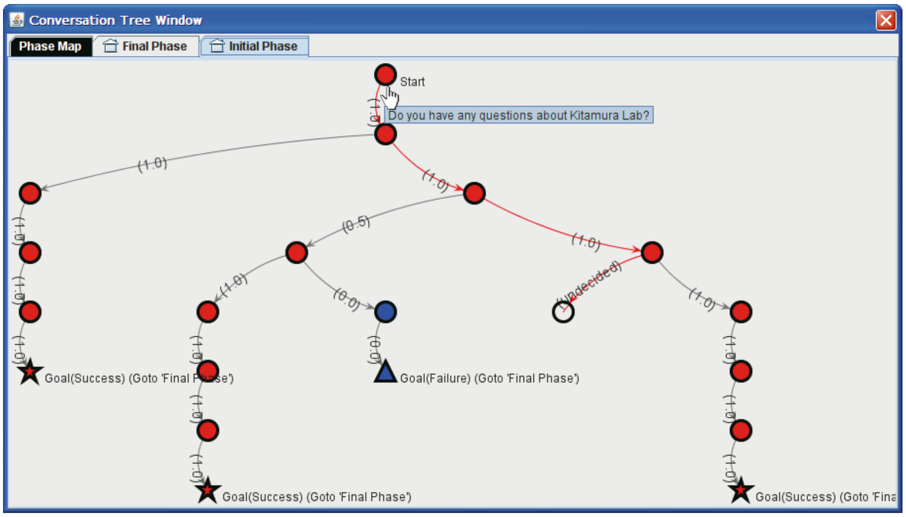

Conversation Tree Window
Phase Map
Final Phase
Initial Phase
Start
Do you have any questions about Kitamura Lab?
(1.0)
(0.5)
(0.0)
(Undecided)
Goal(Success) (Goto 'Final Phase')
Goal(Failure) (Goto 'Final Phase')
Goal(Success) (Goto 'Final Phase')
Goal(Success) (Goto 'Fina

Fig. 10. Displaying a conversation model

Table 1. Specifications of digital cameras A and B

	A	B
Price	¥35,000	¥29,800
Resolution	10M pixels	7M pixels
Weight	154g	131g
Image stabilizer	Yes	No

Table 2. Experiment 1: Result of persuasion

Initial choice	Final choice	Number of participants	Success/Failure
A	A	22(54%)	Failure
	B	19(46%)	Success
B	A	6(30%)	Success
	B	13(70%)	Failure

4. The participant is then asked which camera he/she now prefers. The persuasion succeeds (fails) if he/she changes (does not change) his/her opinion.

5.1 Experiment 1: Input Cost of Wizard

In this experiment, we constructed a conversation model by collecting persuasive conversations between an agent and 60 university students (48 male, 12 female, the average age is 20.9) using the Wizard of Oz method. The results are shown in Table 2. In total, we succeeded in persuading 25 (42%) of the 60 participants.

Figure 11 shows the number of responses made by the agent to each participant. The responses selected by the Wizard are categorized into 4 groups. "CSR (best)" is the context sensitive response with the highest success probability generated by the agent. "CSR (2nd or worse)" covers the context sensitive responses that had 2nd or lower probability generated by the agent. "CFR" covers context free responses. "Wizard input" the responses input by the Wizard. At first, the Wizard has to input most of the responses, but gradually this number falls and the number of responses made by the agent increases. Overall, for the 60 persuasive conversations, the Wizard accepted 602 (48%) of the agent's 1245 responses as appropriate, so this means that the input cost of the Wizard was reduced.

In a remarkable occurrence, the agent succeeded in persuading one participant (no.51) without any input from the Wizard.

5.2 Experiment 2: Persuasiveness of Conversation Model

To determine the persuasiveness of the conversation model described in the previous section, we performed another experiment to persuade 10 students. In this experiment, the agent was limited to one response at each turn. Only when the agent returned no

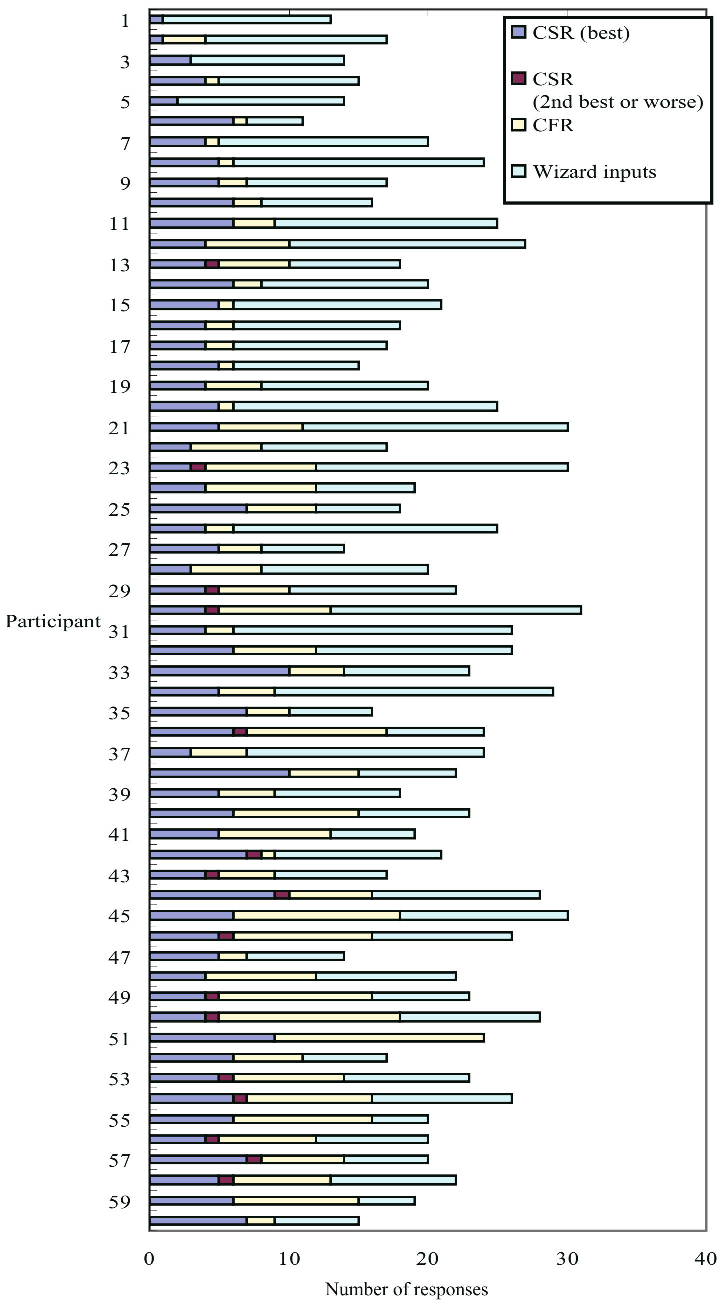


Fig. 11. Experiment 1: Categorized responses

Table 3. Experiment 2: Result of persuasion

Initial choice	Final choice	Number of participants	Success/Failure
A	A	4(57%)	Failure
	B	3(43%)	Success
B	A	0(0%)	Success
	B	3(100%)	Failure

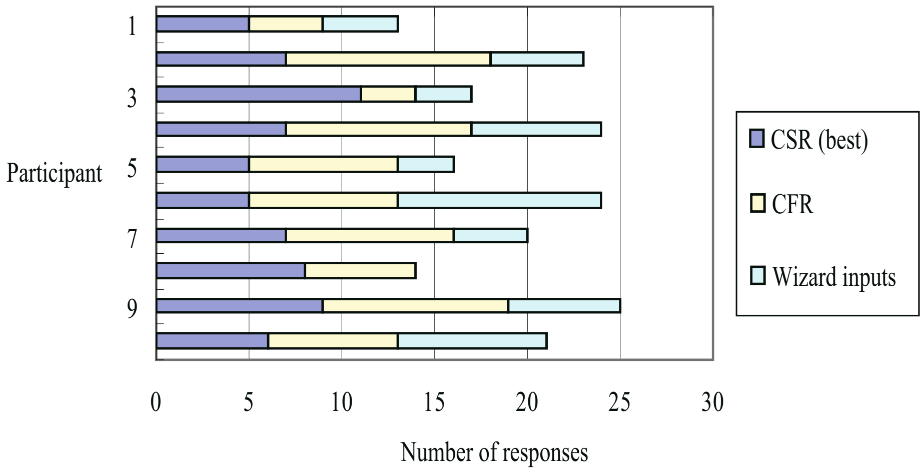


Fig. 12. Experiment 2: Categorized responses

Table 4. The number of generated states of each phase

Phase	Number Of states
Greeting	35
Initial choice	17
A: persuasion	1296
B: persuasion	491
	1839

response, the Wizard input a response as before. The result of this experiment is shown in Table 3. In total, the agent succeeded in persuading 3 (30%) out of the 10 participants.

The responses made by the agent are categorized in Fig. 12. The agent succeeded in persuading one participant (no.8) without any input from the Wizard.

The conversation model created from the two experiments consists of 1839 states as shown in Table 4.

6 Conclusion

Persuasive conversational agents are expected to be virtual shopping clerks on e-shopping sites. To create such agents, we need to create a conversation model that specifies how to reply to inputs from users. In this paper, we proposed an approach to create a conversation model by integrating a learning agent and the Wizard of Oz method. We evaluated the performance of the proposed persuasive conversational agent in a situation where it persuaded users to choose one digital camera over another.

In the 1st experiment with 60 subjects, we could reduce the number of Wizard inputs by 48%, and in the 2nd experiment, the agent that used the conversation model created in the first experiment succeeded in persuading one user (out of 10) without any input from the Wizard; another 2 subjects were persuaded with some assistance by the Wizard.

At present, our persuasive agent requires a lot of assistance from the Wizard to persuade human users. We need to work on to improve the ability of persuasion toward an agent that requires no assistance. In future work, we will improve the success ratio of persuasion. To this end, we need to collect more conversations to create a better conversation model that replies to a larger number of inputs from users. Further work is needed on reducing model redundancy by using natural language processing techniques to handle synonymous sentences.

Another future task is to increase the maintainability of the conversation model. At present, it is not easy to modify the conversation model. We need to develop a GUI for this and to visualize the persuasion strategies contained in the model.

Acknowledgment

This work is partly supported by the Grant-in-Aide for Scientific Research (No.17300050) from Japan Society for the Promotion of Science.

References

1. Fogg, B.J.: Persuasive Technology. Morgan Kaufmann, San Francisco (2003)
2. Cassell, J., et al.: Embodied Conversational Agents. MIT Press, Cambridge (2000)
3. Prendinger, H., Ishizuka, M. (eds.): Life-like Characters. Springer, Heidelberg (2004)
4. Fraser, N.M., Gilbert, G.N.: Simulating Speech Systems. Computer Speech and Language 5(1), 81–99 (1991)
5. Andre, E., Rist, T., Muller, J.: Integrating Reactive and Scripted Behaviours in a Life-Like Presentation Agent. In: Proceedings of the Second International Conference on Autonomous Agent, pp. 261–268 (1998)
6. Andre, E., Rist, T., Muller, J.: WebPersona: A Life-Like Presentation Agent for the World-Wide Web Knowledge-Based Systems 11(1), 25–36 (1998)
7. Stent, A., Dowding, J., Gawron, J.M., Bratt, E.O., Moore, R.: The CommandTalk spoken dialogue system. In: Proc. ACL 1999, pp. 183–190 (1999)
8. Okamoto, M., Yeonsoo, Y., Ishida, T.: Wizard of Oz Method for Learning Dialogue Agents, Cooperative Information Agents V, LNAI 2182. In: Klusch, M., Zambonelli, F. (eds.) CIA 2001. LNCS (LNAI), vol. 2182, pp. 20–25. Springer, Heidelberg (2001)

ASBO: Argumentation System Based on Ontologies

Andrés Muñoz and Juan A. Botía

Department of Information and Communications Engineering
Computer Science Faculty, University of Murcia
Campus de Espinardo, 30100 Murcia, Spain
amunoz@um.es, juanbot@um.es

Abstract. Conflicts are unavoidable in open distributed systems. Belief or semantic conflicts are of special interest in multiagent systems, where agents need to communicate by exchanging knowledge. A common approach to deal with conflicts is the use of argumentation-based negotiation processes. There have been much work in the argumentation research arena. Amongst the outcomes of this research, some generic argumentation frameworks for handling inconsistences can be found, together with several persuasion dialogue systems. The goal of this paper is to contribute in advancing the state-of-art in argumentation by extending the basic mechanisms used in conventional argumentation frameworks. This contribution consists of a new and convenient style of attack to arguments and making explicit the argumentation process structure through an OWL-based ontology. Main benefits of this research are twofold. Firstly, the availability of a more realistic framework thanks to the definition of the new attack. Secondly, to enable automatic reasoning about the argumentation process itself. To illustrate this approach, we expose a persuasive argumentation scenario based on a real situation.

Keywords: Argumentation, ontological reasoning, logic-based systems, conflicts, persuasion dialogue.

1 Introduction

The problem of detection and resolution of conflicts is a constant pattern [9] that repeats when designing open distributed systems. Normally, these systems consist of autonomous entities with their own points of view about the system's current status. A special interest is deserved to potential semantic conflicts [14], or conflicts on the different entities' beliefs. In particular, these conflicts could arise among agents within a MAS (Multi-Agent System) due to each agent possesses a partial view of the whole knowledge maintained in the system.

Briefly, semantic conflicts stem from different pieces of knowledge that are well-founded separately, nevertheless if they are contradictory and all of them occur in the same knowledge base, inconsistencies rise. This conflicting situation gets considerably worse if the knowledge base is built upon any underlying logic, as it leads to the instability of the base, a very undesired situation. By way of example, let us suppose two agents in a MAS devoted to traffic control. $Agent_A$ believes that cars must be forbidden in the city center, according to its own reasons (e.g. cars pollute), and $Agent_B$ holds the contrary opinion, since cars are needed by handicapped people to go to the city center. If we translate these beliefs into some formal language, e.g. First-Order Logic, the result obtained could be represented as $\{forbid(cars)\}$ and $\{\neg forbid(cars)\}$ for agents $Agent_A$ and $Agent_B$, respectively. Each belief is well-founded with respect to the agent that produces it (assuming that each agent's internal knowledge base is consistent). However, if these two agents must live in the same system, the virtually global knowledge base will contain two contradictory statements.

A wide range of techniques have been developed in an attempt to cope with this recurrent problem. Some of them try to avoid conflict occurrence, as for example social laws [26], coordination through cooperation when agents are benevolent [13] and truth maintenance systems [7]. Social laws are difficult to implement in rather simple environments and they seem to be non-applicable in a complex environment like those of open distributed systems. Cooperation implies benevolent agents, but agents in open systems are typically self-interested, or even being willing to cooperate, they pursue their own goals, and it seems hard to define a dynamic cooperation plan needed here. Truth maintenance systems try to explain why conflicts happen by registering the reasons that derive the conflicting conclusions, but they are not able to reason about the conflict itself, i.e., the sources of the conflict are not taken into consideration. On the other hand, negotiation [25] focuses on solving conflicts once they have appeared. Hence, we believe that argumentation-based negotiation processes are needed in open distributed systems to enable effective coordination.

The goal of this paper is intended to advance the state-of-art in argumentation through two different lines of work. Firstly, we specify the abstract argumentation framework stated in [23] by defining an argumentation system based on ontologies. The knowledge that flows through the system is represented by means of Semantic Web ontologies [3](usually expressed in OWL), therefore the underlying logic and the argument modelling will be affected by this decision. Secondly, we try to establish a new style of attack to arguments. This attack is directed to ontology rules (i.e., rules built from ontological classes and its relationships, individuals, etc.) that are used in the argument's premises, by preventing an agent from employing a specific rule and explaining the cause of this attack.

The rest of the paper is structured as follows. Section 2 describes our approach on argumentation systems, ASBO, explaining how to combine ontologies with an argumentation framework. An extension of ASBO including the new attack on ontology rules used in argument construction is presented in Section 3. In section 4, a persuasion scenario in which ASBO is used to build and exchange arguments is described. Section 5 discusses the related work and establishes our contribution to the state of art within argumentation literature. Finally, section 6 summarizes the contribution of this paper and points out the future work.

2 An Argumentation System Based on Ontologies

Following the abstract argumentation framework defined by Prakken [23], there are five basic notions that should be settled to define an argumentation system: an underlying logic language, the concept of argument, the concept of conflict between arguments, a notion of defeat between arguments, and the acceptability status of arguments.

The ASBO approach is based on the usage of Semantic Web OWL-based ontologies ("ontologies" from now on), since it allows an explicit connection between the agent's knowledge representation and the portion of reality that is being represented. Moreover, ontologies allow exchanging and reusing of knowledge thanks to the formal schemes provided. Another important characteristic is the possibility of generating new knowledge from inference processes applied to ontologies. Finally, automatic detection of conflicts is an essential aspect when reasoning with ontologies by means of an inherent validation process.

In order to illustrate how each Prakken's basic notion is defined in ASBO, let us introduce a small example of an scenario formed by two agents. Then, the definition of ASBO follows. For the sake of readability, a language close to First-Order logic is employed here[1], but actually agents' knowledge is represented by using OWL ontologies, as shown in Figure 1. Suppose that agent Ag_1 owns the knowledge base $\Delta_{Ag_1} = (Agent(p), Task(t), mandatory\_task(p,t))$ and the rule set RS_{Ag_1}:

$$\begin{aligned} R_{Urgent} &= (Agent(X), Task(T), mandatory\_task(X,T)) \Rightarrow UrgentTask(T) \\ R_{NoUrgent} &= (Agent(X), Task(T), recommended\_task(X,T), \\ &\quad Overloaded(X)) \Rightarrow TrivialTask(T) \end{aligned}$$

R_{Urgent} classifies a task as urgent if an agent has it annotated as mandatory, whereas rule $R_{NoUrgent}$ states that a task is trivial if its execution is recommended instead of mandatory, and moreover the agent is overloaded. On the other hand, Ag_2 knows $\Delta_{Ag_2} = (Agent(p), Task(t), recommended\_task(p,t))$ with rules RS_{Ag_2}, labeling the task as unnecessary if it is just recommended:

$$R_{Trivial} = (Agent(X), Task(T), recommended\_task(X,T)) \Rightarrow TrivialTask(T)$$

Underlying logic: Description Logic(DL) [2], and in particular OWL-DL [6], is the formalism used in ASBO to represent agents' knowledge. This logic is a subset of First-Order Logics with several restrictions, widely used to represent any domain in a formal and structured way. OWL-DL syntax consists of:

1. A set of unary predicates to represent *concepts* (or *classes*).
2. A set of binary predicates to represent relationships between concepts. These predicates are also known as *properties*.
3. A set of operators to recursively define more complex concepts from other concepts and relationships.

[1] Capital letters denote variables, whereas normal letters are ground literals.

Usually, OWL-DL ontologies are divided into TBox (*terminological*) and ABox (*assertional*) components. The former contains the schema that defines domain concepts and relationships among them, together with a (possibly empty) set of restrictions on the relationships, whereas the latter is populated with instances representing the current situation knowledge, according to the schema. Figure 1 expresses our small example in DL terms. In the TBox, concepts $UrgentTask$ and $Agent$ are given as intersection or union of other concepts as $Person$ or $Task$ together with its properties. For example, $UrgentTask$ is related to a boolean value through the *urgent* relationship. Moreover, $TrivialTask$ is expressed as the complementary concept of $UrgentTask$. In the right side of Figure 1, the Aboxes for agents Ag_1 and Ag_2 are shown, i.e., the Δ of each agent. Both contain the specific and partial world description of each one.

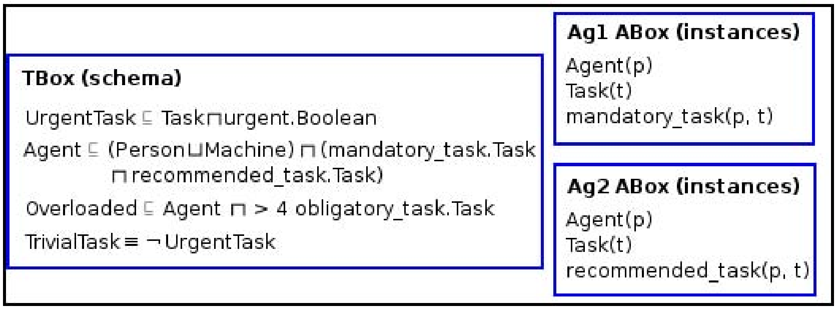

TBox (schema)
UrgentTask ⊑ Task⊓urgent.Boolean
Agent ⊑ (Person⊔Machine) ⊓ (mandatory_task.Task ⊓ recommended_task.Task)
Overloaded ⊑ Agent ⊓ > 4 obligatory_task.Task
TrivialTask ≡ ¬ UrgentTask
Ag1 ABox (instances)
Agent(p)
Task(t)
mandatory_task(p, t)
Ag2 ABox (instances)
Agent(p)
Task(t)
recommended_task(p, t)

Fig. 1. The whole knowledge base for the example. Notice that the TBox is a common schema to both agents, and ABoxes are partial world descriptions for each agent.

As for the ontology rule sets RS, these are expressed with SWRL (Semantic Web Rule Language) [12], an abstract rule language. Rules in SWRL are of the form of an implication among a conjunction of antecedents and a conjunction of consequents, similar to the rules in the example. Both conjunctions consist of *atoms*: unary ($C(x)$) or binary predicates ($P(x, y)$) from the DL language, or *Built-ins*, where C are concepts, P represents relationships, and x, y are variables, instances from ABox or data values. Built-ins are functions that offer different operations on variables (comparison, math,. . .). Thus, all atoms in the antecedent must be true in order to the atoms in the consequents become true.

Arguments: A generic argument structure appears in Figure 2. The main advantage of this representation is to enable automatic detection of conflicts thanks to the ontological formalism and the validation process. The *Argument* concept consists of a *conclusion*, a *support* set, and the argument *state*. The range of values of a state can vary among *acceptable, non-acceptable, conflict* and *unknown*. In case of *non-acceptable* or *conflict* values, the *State* concept allows specifying the *counterarguments* that defeats or conflicts with it. In order to include the new attack over ontology rules (please, see Section 3), the *conclusion* can be either a derived fact supported by the argument, as in most of argumentations systems

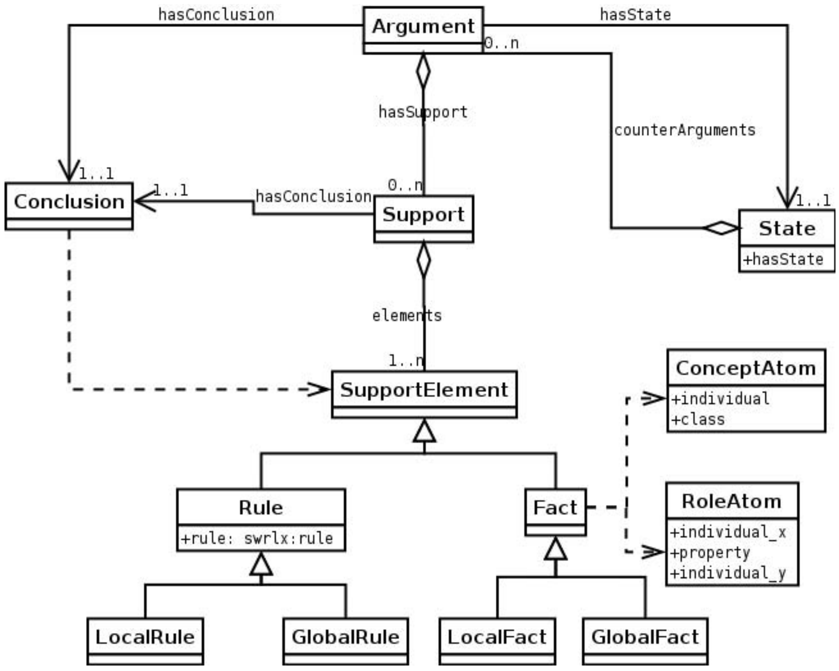


Fig. 2. Argument structure

reviewed in the literature, or an ontology rule. Each element of the *Support* set, S, consists of a conclusion (the intermediate conclusions of an argument) and a *Support Element* set. Each element of the *Support Element* set can be either a *Fact* or a *Rule*. In turn, facts can be divided into *Local Facts*, representing exclusive facts, i.e. facts that a particular agent holds, but the rest of agents do not share or accept. *Global Facts*, in opposition, is the set of beliefs that all agents in a system not only share, but also accept. The fact structure reflects the syntax of Description Logic. As a result, it contains unary predicates (*ConceptAtom*) or binary predicates (*RoleAtom*). Finally, ontology rules are modelled by reusing the SWRL ontology [12]. Here a similar separation as the one that has been made with facts is proposed, but with a subtle difference. Now, a distinction between *global* rules and *local* rules is established. Global rules are those that are shared by all agents in the system, however in this case they can be attacked by more specific rules, the local ones. These are hold exclusively by each agent, and can equally be attacked by other local rules.

This differentiation between non-attackable global real facts and attackable global rules makes sense, because a real fact is something that is true in the system independently of the agents' local knowledge, since it is captured by the system when interacting with its environment. Contrarily, a global rule is introduced by the system designer, and some agents may have more specific rules that defeat them. This idea is somehow similar to DeLP [11], identifying global rules as defeasible, and local ones as strict rules. On the other hand, observe that any element of an argument expressed through an OWL ontology could reside on a different location on the Web. Consequently, arguments can be constructed with different kind of information, from an URI referring to a web page so as to state a Fact, to a web service that implements an argument's conclusion.

In order to build an argument, a reasoning scheme must be taken into account. Following ASBO underlying logic, a deductive inference process is used, in which a conclusion is entailed by means of a set of argument support, S, and an inference operator, denoted as $\vdash$. This operator represents the *modus ponens* inference rule in the deductive process. Returning to the agents and tasks example, the next argument can be derived from Ag_1:

$$\begin{aligned} U_{Urgent} &= \{UrgentTask(t), \langle s_{u_{Urgent},1} \rangle\}, \\ s_{u_{Urgent},1} &= \{Agent(p), Task(t), mandatory\_task(p,t), R_{Urgent}\} \vdash UrgentTask(t), \end{aligned}$$

where $U = \{\phi, \langle S = s_{u,1}, \ldots, s_{u,n} \rangle\}$ is an argument with conclusion ϕ and support set S. $s_{u,i}$ represents the $i-th$ support element for the argument U.

Conflicts: Two main types of conflicts or attacks among arguments have been considered in ASBO: *Semantic conflicts* (these include rebutting and undercutting [18]) and *rule conflicts*. Both conflicts are automatically detected thanks to the validation process that can be run over an ontology.

– *Semantic conflicts*, or "classic" attacks in argumentation, namely *rebut* and *undercut*. Let us suppose two arguments, $A = \{\phi, \langle S_A = s_{A,1}, \ldots, s_{A,n} \rangle\}$, and $B = \{\rho, \langle S_B = s_{B,1}, \ldots, s_{B,n} \rangle\}$. Then, A rebuts B if ϕ and ρ are contradictories (i.e. $\phi \equiv \neg\rho$). Furthermore, A undercuts B if ϕ is inconsistent with any of the B's intermediate conclusions in $s_{B,i}$. Please, see section 3 for a more formal definition in both attacks. These conflicts are related to the semantics of the concepts and relationships defined in the ontology. For example, Ag_1 could generate the previous argument U_{Urgent} inferring t as urgent, whereas another ontology rule may assert that the task t is trivial. This is the $R_{Trivial}$ case. With this rule, Ag_2 could build a counterargument $U_{Trivial}$ that attacks (a rebutting attack) U_{Urgent}, since $UrgentTask$ and $TrivialTask$ are contradictories concepts (please, see Figure 1):

$$\begin{aligned} U_{Trivial} &= \{TrivialTask(t), \langle s_{u_{Trivial},1} \rangle\}, \\ s_{u_{Trivial},1} &= \{Agent(p), Task(t), recommended\_task(p,t), R_{Trivial}\} \vdash TrivialTask(t) \end{aligned}$$

– *Rule conflicts*, this kind of conflict is defined as an attack to any of the ontology rules in an argument by another rule. A special kind of attack could be given not because of incompatible conclusions, as rebutting attack, but due to different rule's antecedents (please, see section 3 for details). For example, Ag_1 could state a counterargument $U_{NoUrgent}$ that conflicts with the argument $U_{Trivial}$'s rule $R_{Trivial}$, because of the rule $R_{NoUrgent}$:

$$\begin{aligned} U_{NoUrgent} &= \{\neg R_{Trivial}, \langle s_{u_{NoUrgent},1} \rangle\}, \\ s_{u_{NoUrgent},1} &= \{R_{NoUrgent}\} \end{aligned}$$

Hence, Ag_1 opposes to the use of $R_{Trivial}$, but it does not oppose to the rule's conclusion, the same in both rules (that task t is trivial). However,

Ag_1 has more specific knowledge (the overloaded status of the agent) to reach that conclusion, and uses it to attack $U_{Trivial}$. As a result, Ag_1 blocks the use of $R_{Trivial}$ and the $TrivialTask(t)$ derivation process is not allowed until $Overloaded(p)$ is proved. Notice that in $U_{NoUrgent}$ the conclusion is an ontology rule. Although just the rule's names have been indicated in the argument, it would actually contain the complete definition of each of them.

Defeat: The relation of defeat between arguments is defined as follows:

*Let U_1, U_2 be two arguments. U_1 **defeats** U_2 iff U_1 is the empty argument and U_2 attacks itself (self-defeating argument), or else if:*

– U_1 undercuts U_2; or
– U_1 rebuts U_2 and U_2 does not undercut U_1.

Moreover, U_1 *strictly defeats* U_2 iff U_1 defeats U_2 and U_2 does not defeat U_1. This definition will be augmented in section 3 after formally defining the ontology rule attack.

Status: An argument can be classified in one of the *acceptable, non-acceptable, conflict* or *unknown* state. To set the status of any argument, it is needed a process that takes into account not only conflicting arguments, but all the relevant arguments. In section 4, a persuasion dialogue system is proposed to establish the status of each argument.

3 Attacking Ontology Rules in Argumentation

The ontology rule attack is part of the ASBO approach. Thanks to the semantic burden that is contained in ontology rules, attacks among these elements can be possible. Here a formal definition of the attack is given. Let us first see how rebutting and undercutting attacks are defined. Using the structure defined in section 2, an argument U is typically formed by a conclusion ϕ and the derivation set S (*Support* set) that support ϕ. In the following example, Propositional Logic is used to derive a from b (initial fact) and the ontology rules $b \to d$ and $d \to a$ (notice that $\vdash$ represents the *modus ponens* inference operator):

$$\begin{aligned} U = \{a, \langle S = s_{u,1}, s_{u,2}\rangle\}&,\\ s_{u,1} = \{b, b \to d\} \vdash d&\\ s_{u,2} = \{d, d \to a\} \vdash a&\end{aligned}$$

"Classic" attacks in argumentation systems are divided into two types: *Rebut*, or to attack the argument's conclusion (i.e. $\{\neg a, \langle \ldots \rangle\}$); and *undercut*, or to attack the intermediate conclusions, i.e. the derived facts in $s_{u,i}$ steps (e.g. $\{\neg d, \langle \ldots \rangle\}$).

Taking a deeper look into the argument structure, it is possible to establish a more specific type of attack on the ontology rules taking part in the s_i steps. For example, an argument $R = \{\neg(b \to d), \langle \ldots \rangle\}$ could be built, which attacks the use of that rule $b \to d$. The consequences derived from this attack have a different semantic meaning than the two classic attacks explained. In this case, the attacker does not necessarily oppose to obtain d, as in the undercutting attack. On the other hand, other reasons for the attack could arise, for instance:

AttI. The opponent might increase the number of conditions to derive d: $b, g \rightarrow d$
AttII. The opponent might have a different set of conditions to derive d: $g \rightarrow d$
AttIII. The opponent might claim that accepting the conditions of the attacked rule makes other rules fire, with undesired consequences for the opponent.

The concept of attacking ontology rules can now be formalized. Let $Ant(r)$, $Cons(r)$ be the set of antecedents and consequents, correspondingly, of a rule r. Let Δ_X be the knowledge base of any agent X. Let RS_X be the ontology rule base of any agent X. These four sets consist of Description Logic conjunctive expressions in which the operations $\cup$ (union), $\cap$ (intersection) and $\equiv$ (equality) are defined as in that logic. Let A, B be two agents, and r_A, r_B two ontology rules, $r_A \in RS_A$ and $r_B \in RS_B$. Then, r_A attacks or conflicts with r_B iff:

Cond1. $\mathrm{Cons}(r_B) \equiv \mathrm{Cons}(r_A)$ and $\exists \alpha \subset \mathrm{Ant}(r_A) \mid \alpha \cap \mathrm{Ant}(r_B) \equiv \emptyset$, or
Cond2. $\exists \delta \subset \mathrm{Ant}(r_B) | \; \delta \equiv \mathrm{Ant}(r_A)$ and $\Delta_A \cup \; \{r_A\} \cup \{\delta\} \vdash \perp$

Condition 1 expresses a conflict between rules with the same consequent, but different sets of antecedents, either because rule r_A contains a greater set of conditionals (AttI., argument $U_{NoUrgent}$ in section 2), or because both sets are totally different (AttII.). Condition 2 defines a conflict due to a subset δ in the r_B's antecedents that fires the rule r_A, entailing an inconsistent state in the A's knowledge base (AttIII.)

Now, the definition of *defeat* between arguments given in section 2 can be expanded with these conditions:

Let U_1, U_2 be two arguments, and r_1, r_2 be two ontology rules. U_1 ***defeats*** *U_2 iff U_1 is the empty argument and U_2 attacks itself (self-defeating argument), or else if:*

- U_1 undercuts U_2; or
- U_1 rebuts U_2 and U_2 does not undercut U_1.
- $\exists r_1 \in U_1, \exists r_2 \in U_2 | r_1$ attacks r_2 according to *Cond1* or *Cond2*.

Moreover, U_1 *strictly defeats* U_2 iff U_1 defeats U_2 and U_2 does not defeat U_1.

4 ASBO in a Persuasive Argumentation Scenario

The argumentation scenario has been developed in a MAS that is able to start discussions on topics proposed by a user, implemented by means of the JADEX[2] platform. There are specialized-knowledge agents for each topic (e.g., environment, health, etc.) depending on the ontology managed by each one. The user poses a question to the system, and the agents try to reach an agreement about the response that should be given, by means of the ASBO that each agent implements and a dialogue game focused on persuasion. For this case of use, a

[2] http://vsis-www.informatik.uni-hamburg.de/projects/jadex/

nowadays argumentative case has been chosen: City council are studying to forbid traffic in the city center. In order to seek out reasons for and against this proposal, the following question is launched to our MAS:

forbid(cars)?

There are two significant specialized agents for this case: an agent *"E"* with a wide knowledge in environmental issues, and agent *"H"* that is specialized in handicapped people topics. Apart from the local knowledge and rules owned by each agent, there is a set of global rules shared by all agents. These rules are intended to represent general behavior, and may be attacked for more specific local rules:

$$\begin{aligned} R_{C1} &= dangerous\_to(x, y) \Rightarrow forbid(x) \\ R_{C2} &= alternative(y, x) \Rightarrow \neg necessary(x) \end{aligned}$$

The first rule states that if an element x represent a risk to y, then the former must be forbidden. R_{C2} establishes that if an element y offers an alternative to x, then the latter loses its "necessary" condition. Now, let us revise the agents' knowledge. Agent E should agree with forbidding cars, as its goal is to protect the environment. E owns an environment ontology, instantiated with the following set of local facts and rules:

$$\begin{aligned} \Delta_E &= \{emit(cars, fumes), threaten(cars, pedestrian), alternative(bus, car)\} \\ R_{E1} &= emit(x, fumes) \Rightarrow pollute(x) \\ R_{E2} &= emit(x, noise) \Rightarrow pollute(x) \\ R_{E3} &= threaten(x, z) \Rightarrow dangerous\_to(x, z) \\ R_{E4} &= pollute(x) \Rightarrow forbid(x) \end{aligned}$$

As counterpart, H tries to watch over a bigger comfort for handicapped people, therefore it will oppose to forbid cars since it is a valid mean of transport that minimizes their problems. H instantiates a handicapped ontology in the following way:

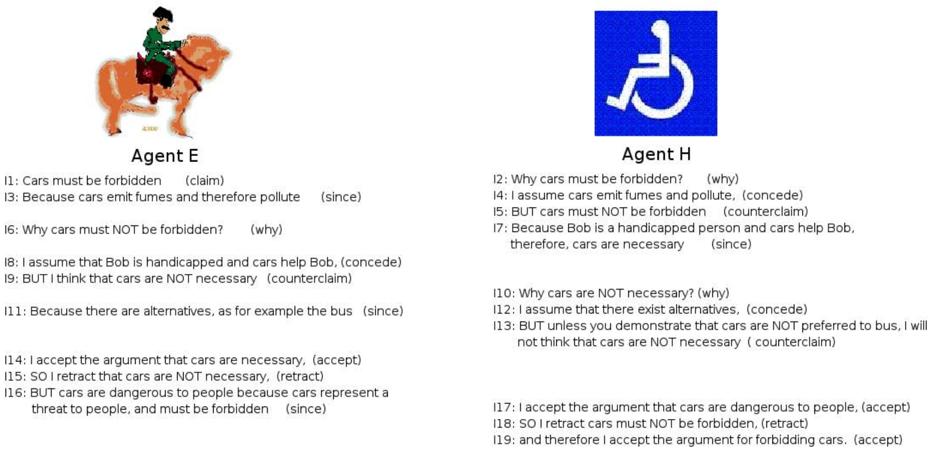
$$\begin{aligned} \Delta_H &= \{handicapped(Bob), help(cars, Bob), preferred\_to(car, bus)\} \\ R_{H1} &= handicapped(z), help(x, z) \Rightarrow necessary(x) \\ R_{H2} &= alternative(y, x), \neg preferred\_to(x, y) \Rightarrow \neg necessary(x) \\ R_{H3} &= necessary(x) \Rightarrow \neg forbid(x) \end{aligned}$$

The fact $preferred\_to(car, bus)$ means that H believes that cars are preferred to buses. R_{H1} states that an element x that helps a handicapped person z has the quality of being necessary. It can be observed that R_{H2} is a specialization of rule R_{C2}, since agent H considers that an element x is not necessary if an alternative y does exist, and moreover that x is not preferred to the alternative y. Supposing that H is a rational agent, it will prefer R_{H2} to R_{C2} in an inference process.

Once the agents have been defined, the next step is to set up a mechanism to reach a consensus about the answer to the question launched in the system. For that objective, agents must participate in a dialogue in which each agent tries to persuade the others that its proposal is the most reasonable, and in consequence it must be chosen by the system. The dialogue system consists of three components:

– A communication language, C, that is known by all agents in the system.
– A protocol or dialogue rules that ensures the ending of the dialogue and fairness among agents.
– A logic language, L, to express the content of the messages exchanged between agents (arguments).

The ASBO dialogue system is based on [21], with the difference that L belongs to Description Logic. The TBox of each ontology used in the dialogue must be accessible to all the participants, in order to understand and check the consistency of the local knowledge exchanged in the arguments. Moreover, we expand the dialogue system to include the ontology rule attack of section 3.



I1: Cars must be forbidden (claim)
I3: Because cars emit fumes and therefore pollute (since)

I6: Why cars must NOT be forbidden? (why)

I8: I assume that Bob is handicapped and cars help Bob, (concede)
I9: BUT I think that cars are NOT necessary (counterclaim)

I11: Because there are alternatives, as for example the bus (since)

I14: I accept the argument that cars are necessary, (accept)
I15: SO I retract that cars are NOT necessary, (retract)
I16: BUT cars are dangerous to people because cars represent a threat to people, and must be forbidden (since)

I2: Why cars must be forbidden? (why)
I4: I assume cars emit fumes and pollute, (concede)
I5: BUT cars must NOT be forbidden (counterclaim)
I7: Because Bob is a handicapped person and cars help Bob, therefore, cars are necessary (since)

I10: Why cars are NOT necessary? (why)
I12: I assume that there exist alternatives, (concede)
I13: BUT unless you demonstrate that cars are NOT preferred to bus, I will not think that cars are NOT necessary (counterclaim)

I17: I accept the argument that cars are dangerous to people, (accept)
I18: SO I retract cars must NOT be forbidden, (retract)
I19: and therefore I accept the argument for forbidding cars. (accept)

Fig. 3. Sequence of a persuasion dialogue

The dialogue listed below is an example of a possible complete dialectical process in this scenario. Notation $X.I_{num}$ means that agent X utters the illocution *Num*. In figure 3 the same dialogue is shown in a face-to-face way. Let us explain the steps that constitute it:

$E.I_1$: **claim**(*forbid(cars)*)
$H.I_2$: **why**(*forbid(cars)*)
$E.I_3$: $forbid(cars)$ **since**(U_p)
$H.I_4$: **concede**(*emit(cars, fumes), pollute(cars)*)
$H.I_5$: **counterclaim**($U_p, \neg forbid(cars)$)
$E.I_6$: **why**($\neg forbid(cars)$)
$H.I_7$: $\neg forbid(cars)$ **since**($U_{\neg p}$)
$E.I_8$: **concede**(*handicapped(Bob), help(cars,Bob)*)
$E.I_9$: **counterclaim**($U_{\neg p}, \neg necessary(cars)$)
$H.I_{10}$: **why**($\neg necessary(cars)$)

$E.I_{11}$: $\neg necessary(cars)$ **since**($U_{\neg n}$)
$H.I_{12}$: **concede**(*alternative(bus,car)*)
$H.I_{13}$: **counterclaim**($U_{\neg n}, R_{H2}$)
$E.I_{14}$: **accept**($U_{\neg n}, R_{H2}$)
$E.I_{15}$: **retract**($\neg necessary(cars)$)
$E.I_{16}$: $forbid(cars)$ **since**(U_a)
$H.I_{17}$: **accept**(U_a, *dangerous_to(cars,pedestrians)*)
$H.I_{18}$: **retract**($\neg forbid(cars)$)
$H.I_{19}$: **accept**(U_p , *forbid(cars)*)

I_1: Agent E tries to obtain the conclusion $forbid(cars)$, by looking into its own knowledge first. E notices that rule R_{E4} supports *forbid(cars)* if the antecedent is instantiated with *pollute(cars)*. By means of R_{E1} or R_{E2}, it could be possible. As E knows that *emit(cars,fumes)*, R_{E1} can be triggered. Thus, the following argument is built:

$$\begin{aligned} U_p = \{&forbid(cars), \langle s_{p_1}, s_{p_2}\rangle\},\\ &s_{p_1} = \{emit(cars, fumes, R_{E1})\} \vdash pollute(cars),\\ &s_{p_2} = \{pollute(cars), R_{E4}\} \vdash forbid(cars)\end{aligned}$$

After building the argument, E launches its conclusion into the system.

I_2: Agent H receives the E's illocution. Since its role drives him against forbidding cars, H questions E's claim.

I_3: E exposes the reasons of its claim in I1, by communicating the argument that supports it.[3]

I_4, I_5: H needs now to attack argument U_p. Since H does not possess any knowledge that counterattacks the facts *emit(cars, fumes)* and *pollute(cars)*, nor the argument's rules, it must assume these propositions. Then, H focuses its attack on the U_p conclusion, i.e. H tries to build an argument for $\neg forbid(cars)$. Looking into its local knowledge, by taking rules R_{H1} and R_{H3} together with facts *handicapped(Bob)* and *help(cars, Bob)*, the argument $U_{\neg p}$ is built:

$$\begin{aligned} U_{\neg p} = \{&\neg forbid(cars), \langle s_{\neg p_1}, s_{\neg p_2}\rangle\},\\ &s_{\neg p_1} = \{handicapped(Bob), help(cars, Bob), R_{H1}\} \vdash necessary(cars),\\ &s_{\neg p_2} = \{necessary(cars), R_{H3}\} \vdash \neg forbid(cars)\end{aligned}$$

After building this argument, H communicates that assumes the premises and rules of the previous argument (I_4), however it rebuts the conclusion (I_5).

I_8, I_9: It is now the turn of E to attack argument $U_{\neg p}$. Firstly, the agent must accept the *handicapped(Bob)* and *help(cars, Bob)* premises, because there is not knowledge against them. About the *necessary(cars)* premise, E does not have any specific knowledge against it either, but taking the global rules into account,

[3] Steps (I10, I11) are similar to this last pair (I2, I3), and the same for (I6, I7) but with agent E asking for reasons, so they will not be explained again.

R_{C2} allows E to attack that premise if *alternative(y, car)* could be justified. As *alternative(bus, car)* is hold by E, the following argument undercuts $U_{\neg p}$:

$$\begin{aligned} U_{\neg n} &= \{\neg necessary(cars), \langle s_{\neg n_1} \rangle\}, \\ s_{\neg n_1} &= \{alternative(bus, car), R_{C2}\} \vdash \neg necessary(cars) \end{aligned}$$

I_{12}, I_{13}: After assuming the fact *alternative(bus, car)*, H detects that one of their specific local rules, R_{H2}, has the same consequent that R_{C2}. Furthermore, this rule adds a new particular condition to obtain the conclusion $\neg necessary(cars)$, namely that the element which an alternative is offered to must not be preferred to this alternative. As a result, R_{H2} attacks R_{C2} according to the *AttI* definition in section 3. Now H attacks argument $U_{\neg n}$ by defeating rule R_{C2} with rule R_{H2}:

$$U_{\neg R_{C2}} = \{\neg R_{C2}, \langle R_{H2} \rangle\}$$

H communicates this attack by means of a *counterclaim* to argument $U_{\neg n}$ including rule $R_{H2}(I_{13})$.

I_{14}, I_{15}, I_{16}: E's argument $U_{\neg n}$ is attacked by R_{H2}. In order to maintain this argument, E needs to justify that cars are not preferred to buses. Nevertheless, this knowledge is not derivable from the agent's knowledge base, nor from global rules. Hence, E accepts argument $U_{\neg p}(E.I_{14})$, in response to $H.I_{13}$ and retracts $\neg necessary(cars)(E.I_{15})$, in response to $H.I_{10}$. Here it is important to notice that although E accepts argument $U_{\neg p}$, the dialogue has not finished, since E still holds its argument U_p, which is in a rebuttal conflict with $U_{\neg p}$ now. This agent has run out of local knowledge to build a new attack or argument for *forbid(cars)*. However, looking into global rules, E notices that R_{C1} opens a new way to support its goal, by justifying *dangerous_to(cars, y)*. Using now its specific knowledge, R_{E3} and the fact *threaten(cars, pedestrian)*, E derives U_a:

$$\begin{aligned} U_a &= \{forbid(cars), \langle s_{a_1}, s_{a_2} \rangle\}, \\ s_{a_1} &= \{threaten(cars, pedestrian), R_{E3}\} \vdash dangerous\_to(cars, pedestrian), \\ s_{a_2} &= \{dangerous\_to(cars, pedestrian), R_{C1}\} \vdash forbid(cars) \end{aligned}$$

Notice that this illocution $(E.I_{16})$ is a new response to $H.I_7$, since it attacks argument $U_{\neg p}$.

I_{17}, I_{18}, I_{19}: Finally, H accepts argument U_a, because there is not any new knowledge or rules that can attack it. Since this argument defeats $U_{\neg p}$, it is not anymore a valid argument. Thus, H retracts $\neg forbid(cars)$ in response to $E.I_6$. Because H is not able to make any more movements, the argument U_p is accepted. As a result, the conclusion *forbid(cars)* is also accepted, which is the response given to the user eventually.

5 Related Work

Argumentation deals with several fields in knowledge engineering [5]. Nonmonotonic and Defeasible Reasoning has created an extensive line of work, of which some of the most remarkable publications are Dung [8], Bondarenko [4] and Prakken [20]; the latter specially focused on legal reasoning. As a result, an abstract and generic framework for reasoning under incomplete and inconsistent information has been defined [23]. Expanding the scope of argumentation, Fox and Krausse [10] propose applying it to the problem of decision making under uncertainty. In this approach, the abstract argumentation framework must be extended with representation of the agents' values, beliefs and preferences, and on the other hand, the process of decision making is complicated by uncertainty on the information. Finally, argumentation has been considered in distributed settings, in particular in multi-agent systems, in order to achieve acceptable agreements between agents [17]. In this way, a negotiation protocol is defined via argumentation [16], that leads to a persuasion dialogue in which an agent tries to convince others that its conclusion is the most acceptable [1].

One of the possibilities offered by ASBO is to define and implement a type of attack on rules from the argument's premises. The idea of attacking rules from an argument is not new. It can also be found in OSCAR [19] and in [24]. However, they do not explain how that attack could be developed. Here, a definition and implementation of a rule attack is introduced, taking into account both the rule format and its semantic representation.

There exist several formal systems for an argumentation persuasion dialogue [22]. The Toulouse-Liverpool approach [15] consists of a two-party dialogue, with a communication language based on claims, challenges, concessions and questions, but without a explicit reply structure. Propositional logic is used as a language to represent arguments. In this system is difficult to have dialogues where arguments for and against the same claim are exchanged. Another approach, the Prakken's framework [21], allows specifying two-party persuasion dialogues which status is defined exploiting a tree structure that is built as the dialogue progresses. Termination is defined as the situation that a party is to move but has got no legal moves. The abstract protocol proposed is multi-move and multi-reply, and allows for all kinds of instantiations. Prakken's approach has been used to define a persuasion dialogue within ASBO, extending the protocol with new illocutions in order to include rule attacks.

6 Conclusions and Future Work

Argumentation has proved to be an efficient and useful mechanism to deal with semantic or belief conflicts. In this paper, we pursue to advance the state-of-art in this field by defining a specific type of attack between arguments through ontology rules. As a result, an argumentation framework that questions the reasoning process itself is enabled, extending the possible argumentation lines in this framework. Moreover, the combination of Semantic Web ontology techniques with an abstract argumentation framework is accomplished to obtain an argumentation system based on ontologies, in a first step to automatically detect and resolve conflicts. By representing arguments and their relations by means of OWL-based ontologies, it is viable to take advantage of the explicit semantic information that is contained in this model. Because of the fact that an ontological formalism is used in our argumentation system, the underlying logic, the concept of argument, conflict and the defeat relation between arguments have been properly adapted. To illustrate the new style of attack and the argumentation system, an example based on a persuasion scenario has been developed.

Our efforts are directed to define and validate a dialogue system that fits into our argumentation system, including new illocutions to adapt the new attack and arguments as ontology instances. On the other hand, definition of conflicts and defeat concepts by an ontology model is also being studied, in order to complete our semantic model of argumentation. Finally, another future goal is the automatic solving of conflicts by means of "ad-hoc" inference rules defined by agents themselves.

Acknowledgments. This work has been supported by the Fundación Séneca grant "Programa de Ayuda a los grupos de excelencia 04552/GERM/06" and thanks to the Spanish Ministerio de Ciencia e Innovación (MICINN) under the grant AP2006-4154 in frames of the FPU Program.

References

1. Amgoud, L., Parsons, S.: Agent dialogues with conflicting preferences. In: Meyer, J.-J.C., Tambe, M. (eds.) ATAL 2001. LNCS (LNAI), vol. 2333, pp. 190–205. Springer, Heidelberg (2002)
2. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic handbook: theory, implementation, and applications. Cambridge University Press, New York (2003)
3. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (2001)
4. Bondarenko, A., Dung, P.M., Kowalski, R.A., Toni, F.: An abstract, argumentation-theoretic approach to default reasoning. Artificial Intelligence 93(1-2), 63–101 (1997)
5. Carbogim, D.V., Robertson, D., Lee, J.: Argument-based applications to knowledge engineering. Knowledge Engineering Review 15(2), 119–149 (2000)
6. Dean, M., Connoll, D., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: Web ontology language (OWL). Technical report, W3C (2004)
7. Doyle, J.: A truth maintenance system. Artificial Intelligence 12, 231–272 (1979)
8. Dung, P.M.: On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. Artificial Intelligence 77(2), 321–357 (1995)
9. Dunlop, N., Indulska, J., Raymond, K.: Methods for conflict resolution in policy-based management systems. In: EDOC 2003: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing, p. 98. IEEE Computer Society Press, Washington (2003)
10. Fox, J., Krause, P., Ambler, S.: Arguments, contradictions and practical reasoning. In: ECAI 1992: Proceedings of the 10th European conference on Artificial intelligence, pp. 623–627. John Wiley & Sons, Inc., New York (1992)
11. García, A.J., Simari, G.R.: Defeasible logic programming: an argumentative approach. Theory Practice Logic Programmation 4(2), 95–138 (2004)

12. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. Technical report, W3C (2004)
13. Lesser, V.: Cooperative multiagent systems: A personal view of the state of the art. IEEE Transactions on Knowledge and Data Engineering 11(1) (January 1999)
14. Munoz, A., Botia, J.A., Garcia, F.J., Martinez, G., Skarmeta, A.F.G.: Solving conflicts in agent-based ubiquitous computing systems: a proposal based on argumentation. In: Agent-Based Ubiquitous Computing (ABUC) Workshop, International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii (May 2007)
15. Parsons, S., McBurney, P.: Argumentation-based communication between agents. In: Communication in Multiagent Systems, pp. 164–178 (2003)
16. Parsons, S., Sierra, C., Jennings, N.R.: Agents that reason and negotiate by arguing. Journal of Logic and Computation 1998(3), 261–292 (1998)
17. Parsons, S.D., Jennings, N.R.: Negotiation through argumentation-A preliminary report. In: Proceedings of the Second International Conference Multi-Agent Systems (ICMAS 1996), Kyoto, Japan, pp. 267–274 (1996)
18. Pollock, J.L.: Cognitive Carpentry: A Blueprint for how to Build a Person. MIT Press, Cambridge (1995)
19. Pollock, J.L.: Rational cognition in OSCAR. In: ATAL 1999: 6th International Workshop on Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL), pp. 71–90. Springer, Heidelberg (2000)
20. Prakken, H.: From logic to dialectics in legal argument. In: ICAIL 1995: Proceedings of the 5th international conference on Artificial intelligence and law, pp. 165–174. ACM Press, New York (1995)
21. Prakken, H.: Coherence and flexibility in dialogue games for argumentation. J. Log. and Comput. 15(6), 1009–1040 (2005)
22. Prakken, H.: Formal systems for persuasion dialogue. Knowledge Engineering Review 21(2), 163–188 (2006)
23. Prakken, H., Sartor, G.: A dialectical model of assessing conflicting arguments in legal reasoning. Artificial Intelligence and Law 4(3-4), 331–368 (1996)
24. Prakken, H., Vreeswijk, D.: Logical Systems for Defeasible Argumentation. Handbook of Phil. Logic, vol. 4, pp. 219–318. Kluwer Academic Publishers, Dordrecht (2002)
25. Rosenschein, J.S., Zlotkin, G.: Rules of Encounter.Designing Conventions for Automated Negotiation among Computers. MIT Press, Cambridge (1994)
26. Shohama, Y., Tennenholtz, M.: On social laws for artificial agent societies: off-line design. Artificial Intelligence 73, 231–252 (1995)

Controling Contract Net Protocol by Local Observation for Large-Scale Multi-Agent Systems

Toshiharu Sugawara[1], Toshio Hirotsu[2], Satoshi Kurihara[3], and Kensuke Fukuda[4]

[1] Dept. of Computer Science and Engineering, Waseda University, Tokyo 1698555, Japan
[2] Dept. of Information and Computer Sciences, Toyohashi University of Technology
[3] Inst. of Scientific and Industrial Research, Osaka University
[4] National Institute of Informatics, Chiyoda, Tokyo 100-000, Japan

Abstract. We describe a new adaptive manager-side control policy for the contract net protocol that uses the capabilities of all agents in a massively multi-agent system (MMAS). Recent advances in Internet services, pervasive computing, and grid computing require sophisticated MAS technologies to effectively use the large amount of invested computing resources. To improve overall performance, tasks must be allocated to appropriate agents, and from this viewpoint, a number of negotiation protocols were proposed in the MAS context. Most assume a small-scale, unbusy environment, however. We previously reported the possibility that, using contract net protocol (CNP), the overall efficiency improved by an adequate control of degree of fluctuation in the awarding phase, when the MMAS is in specific states. In this paper, we propose the method to estimate these specific states from the bid values, which have hitherto not been used effectively. Then the new manager-side policy flexibly and autonomously introduces some degree of fluctuation responsive to the estimated states. We also demonstrate that our proposed CNP policy provides considerably better performance than naive CNP and CNP with inflexible policies, even though our policy does not use global information.

1 Introduction

Although recent advances in Internet services, sensor networks, pervasive computing, and grid computing exhibit the need for multi-agent systems (MAS), they further require more sophisticated MAS technologies for large-scale and busy environments. For example, e-commerce transactions, which frequently appear in the current Internet era, consist of coordinated tasks including interactions among a variety of information agents in charge of customer authentication and management, stock management, shipping control, and payment processing. These kinds of tasks simultaneously and frequently occur throughout in the world. In sensor-network applications, information agents reside in many sensor and computational devices, and many computational entities in grid computing should concurrently process the assigned subproblems into which a large computation problem is decomposed. In these applications, many of the tasks should be allocated appropriately for efficient, high-quality services. Of course, providers of these services on the Internet want to maximally utilize the devices and equipment in which they have invested and, at the same time, users want comfortable, quick responses from these systems. Thus, task allocation in massively multi-agent systems (MMAS) is a central issue for fully utilizing the potential performance of all agents deployed in different locations.

In the context of MAS, research in this line is of great concern, and a number of negotiation protocols, such as such as the contract net protocol (CNP) [1,2], multi-stage negotiation protocol [3], and a variety of auction protocols [4,5], have been proposed. In particular, CNP and its extensions have been widely used in certain applications because of the simplicity and good performance of CNP [2,6]. In CNP, an agent plays one of two roles: *managers* are responsible for allocating tasks and monitoring processes, while *contractors* are responsible for executing the allocated tasks. A manager agent announces a task to the contractor agents, which bid for the task with certain promised values or prices (such as cost, duration, and payment). The manager agent then awards the task to the contractor (the *awardee*) that bid with the best value and allocates the task to it.

It is obvious that interference among agents is often observed in this kind of negotiation protocol if many managers have tasks to allocate to efficient contractors. In basic CNP, a contractor agent receives task announcements one by one. When many tasks are announced by many managers, however, they have to wait a long time to receive a sufficient number of bids. In the original conception of CNP [1], the use of multiple bids was proposed for concurrently handling many announcements. If a contractor is awarded multiple tasks simultaneously, however, it may not be able to provide the quality or performance promised. In fact, the more highly capable contractor agents are selected as awardees by many manager agents, leading to a concentration of tasks. In addition, a large number of tasks in a MMAS induce an excessive number of messages, which make all agents busy with (1) reading and analyzing the many messages received (all agents), (2) deciding whether to bid for announced tasks (contractors), (3) calculating bid values (contractors), and (4) selecting awardees (managers). This degrades the overall performance of an MAS.

A simple solution to this problem is to implement manager-side control by restricting announcements to certain selected agents to reduce the number of messages and simplify the award process. In this paper, we call this approach *restricted CNP*. However, strong restrictions may also degrade the performance of task, because they may not be announced to idle and/or highly capable agents. It is unclear whether the overall performance of MAS will ultimately improve or worsen if tasks are more widely announced, especially in an MMAS environment in which more than thousands of agents interact with one another. A number of papers e.g., [7,2,8] have proposed restricting the audience for task announcement to improve performance, especially to avoid message congestion or message sending to uninterested agents, for small-scale MASs. To the best of our knowledge, however, there has been little research on the efficiency and effectiveness of CNP (or more generally, negotiation protocols including CNP) when applied to an MMAS.

The first goal of our research is to understand the behavior of CNP in an MMAS to enable development of efficient, large-scale negotiation protocols. Toward this goal, we previously investigated the performance of an MMAS, especially the overall efficiency and reliability of contracted bid values, when tasks were allocated by CNP with a variety of manager-side controls such as announcement restriction [9,10][1]. These papers indicates the possibility that appropriate degree control of fluctuation in the award selection and appropriate control of announcement restriction in the announcement phase on the basis of the MMAS task load greatly improve overall performance. Note that "fluctuation" in the award selection means that managers do not always select the best contractor among bidding contractors.

Accordingly, we have now developed a flexible manager-side control policy, including announcement and award phases for CNP, that improves overall system performance, using these results. First, we designed the control of fluctuation in award selection on the basis of the system's task load of and confirmed that this control policy could considerably improve overall performance. In actual open-system environment, however, it is almost impossible to acquire global information such as the task load of an entire MMAS. We thus modified this control policy so that each manager can estimate the task loads of its local contractors in accordance with the received bid values (with or without supplemental data), which were previously used only to select awardees. The performance under this flexible control policy was compared with that under other policies that either have no flexibility or that have some degrees of flexible fluctuation based on the global information about the task load. Our new policy outperformed both types, even though it does not use global information.

In this paper, we first describe the restricted CNP model and the simulation model. Then, we summarize our previous findings[9,10] for how degree of fluctuation in the award phase affects the overall efficiency of MMAS under CNP-based task allocation and execution. Next we introduce manager-side award control using the global information about the task load. We then describe how this control is modified to estimate the task load states without using global information. Finally, we discuss the results of the simulation and compare the performance under our control policies with that under other policies.

2 Simulation

2.1 Restricted CNP Model

Let $A = \{a_1, \ldots, a_n\}$ be a set of agents, $M = \{m_j\}(\subset A)$ be a set of managers that allocate tasks, and $C = \{c_k\}(\subset A)$ be a set of contractors that can execute allocated tasks if a contract is awarded. We assume that $M \cap C = \emptyset$ and $A = M \cup C$.

When manager m_j has task T, it allocates T to another agent in accordance with CNP. First, m_j announces task T to all contractors in C (i.e., the announcement phase). For brevity, we assume that all contractor agents can execute T. A contractor receiving this announcement must decide whether to bid for this task. If it decides to bid, it sends m_j a bid message with a certain value called the *bid value* (i.e., the bidding phase). Although bid values in general might include parameters such as the price for executing T, the quality of the result, and a combination of these values, timely responses are always of great concern in interactive services and real-time applications. Thus, we

[1] Ref. [10] has been submitted but has not been accepted yet, so is not available; its major results are shown in Fig. 1.

assume that all agents are rationally self-interested on the basis of efficiency and that their bid values are simply promised times for completing T. Finally, m_j selects a contractor, usually one that bid the best value, and sends an award message to the awardee allocating the announced task (i.e., the award phase).

As the basic CNP [1] has only been used for small-scale MASs in non-busy environments, we extend it to busier, MMAS environments. First, as done previously [1], we assume that contractors are allowed to submit multiple bids concurrently in response to task announcements from multiple managers so that we can apply this approach to busy large-scale applications. Second, unlike in the original CNP, two new CNP messages, *regret* and *no-bid* messages, are introduced (e.g., [2,11]). Regret messages are sent in the award phase to contractors that have not been awarded the contract, while no-bid messages are sent to managers when contractors decide not to bid on an announced task. Using these messages avoids long waits for bid and award messages.

Next, we define restricted CNP. First, let us assume that $|A|$ is large (on the order of thousands), so $|M|$ and $|C|$ are also large, and that the agents are distributed widely, like servers and customer agents on the Internet. For $m_j \in M$, let K_{m_j} be a set of contractors known to m_j; m_j can only announce a task to contractors in K_{m_j}. Set K_{m_j} is called the *scope* of manager m_j. Restricted CNP is thus defined as CNP in which (1) multiple bids and regret and no-bid messages are allowed and (2) each manager m_j can announce tasks to only those contractors in K_{m_j} selected by a certain policy, called the *announcement policy*. Hereafter, the set of contractors selected according to the announcement policy is called the *audience*. We believe that an appropriate announcement policy can reduce the total number of messages and the cost of announcing bid and award decisions, thus improving overall performance.

2.2 Simulation Model

We set $|C| = 500$ and $|M| = 10000$ in our simulation model[2]. The agents are randomly placed on the points of a 150 x 150 grid with a torus topology. Then, the Manhattan distance *dist*(a_i,a_j) between agents a_i and a_j is defined on this grid. Using this distance, we set the communication cost (or delay) for messages from a_i to a_j. This cost is denoted by *cost*(a_i,a_j). The communication cost ranges between 1 and 14 (in *ticks*, the unit of time in the simulation), in proportion to the distance, *dist*(a_i,a_j). The elements of K_{m_j} for $\forall m_j \in M$ are also defined according to this distance; they consist of the nearest 50 contractors to m_j. More precisely, for integer $n > 0$, let $K_{m_j}(n) = \{c \in C|$ *dist*(m_j,c) $\leq n\}$. It follows that $K_{m_j}(n) \subset K_{m_j}(n+1)$. K_{m_j} is defined as the smallest $K_{m_j}(n)$, such that $|K_{m_j}(n)| \geq 50$. Set K_{m_j} remains static once it is calculated. Note that this grid is not introduced for the inter-agent structure but is done for defining the distance between agents; this structure is determined by the scopes of all agents.

With every tick, tl tasks on average are generated, based on a Poisson distribution, in the simulation environment and randomly assigned to different tl managers, where tl is a positive number. Parameter tl is called the *task load* and denotes tl tasks per tick, or simply tl T/t. A manager assigned a task immediately initiates restricted CNP

[2] We assume that the contractor agents run on the Internet, providing services requested by manager agents, which correspond to clients.

to allocate the task to an appropriate contractor. Note that previously [9,10], tasks were constantly generated every tick. Although this may produce slightly different data than previously, the difference is negligible.

For task T and agent a_i, we introduce two parameters: the associated cost of T, *cost*(T), expressing the cost to complete T, and the ability of a_i, a(a_i), expressing the processing speed of a_i. For convenience, we adjust these parameters so that contractor c_i can complete T in *cost*$(T)/a(c_i)$ ticks. Since our experiments were designed simply to clarify the performance of restricted CNP in an MMAS, we assumed that all tasks would have the same cost, i.e., 2500. The abilities of the contractors were initially assigned so that the values of *cost*$(T)/a(c_i)$ (where $i = 1, \ldots, 500$) were *uniformly distributed* over the range $20 - 100$; this means that the values of a(c_i) range from 25 to 125.

When contractor c_i is awarded a task, c_i immediately executes it if it has no other tasks. If c_i is already executing another task, the new task is stored in c_i's queue, which can hold up to 20 tasks. The tasks in the queue are then executed in turn. Tasks that cannot be stored because of a full queue are dropped.

The bid value reflecting the state of contractor c_i is the expected response time, calculated as follows. Suppose that s tasks are queued in c_i. As c_i's bid value is $s * (2500/a(c_i)) + \alpha$, where α is the required time to complete the current task, smaller bid values are better. In multiple bidding, c_i might have a number of uncertain bids for which results have not yet been received. These bids are not considered, however, because it is uncertain whether they will be awarded. This means that contractors always submit bids when a task announcement arrives. Note that although we can introduce a bidding control to avoid over-allocation, by disallowing multiple bids when a contractor is busy, we do not consider this kind of contractor-side control here. The use of other bidding controls will be discussed elsewhere.

The *completion time* for each task is the elapsed time observed by the manager, from the time an award message with the allocated task was sent to the time a message indicating that the task has been completed is received. The completion time thus includes the communication time in both directions, the queue time, and the execution time[3]. We define the *overall efficiency of MAS* as the average completion time observed for all managers and the *reliability* as the expected value of the differences between the completion times and the promised response times. We can assume that a smaller expected value of $\{d_{c_i}\}_{c_i \in C}$ and a smaller standard deviation indicate higher reliability of the MAS, where d_{c_i} is the difference between the bid value of contractor c_i and the completion time of the task by c_i.

The simulation data reported here are the mean values from three independent experiments using different random number seeds. The theoretical limit of processing capability, that is, the cumulative capability of all contractors of the MAS, in the three experiments ranged from 9.7 to 10.2 T/t, with an average value of 9.9 T/t. We set parameter tl to 0.1, 0.5–1, 3–6, 9-10, or 11; the conditions for the MAS in each case are listed in Table 1.

[3] Because our goal is to clarify the performance of MMAS, the costs of processing announcement messages and selecting a bid from bid messages have been not included in the completion time.

Table 1. MAS conditions for various task load

tl (T/t)	Condition
0.1	Task load is extremely low; multiple bids are rare.
0.5–3	MAS is not busy.
3–6	MAS is moderately busy; no tasks are dropped.
9–10	Task load is near limit of MAS cumulative capability; some tasks may be dropped.
11	MAS is extremely busy, beyond the theoretical limit of all contractors; many tasks are dropped.

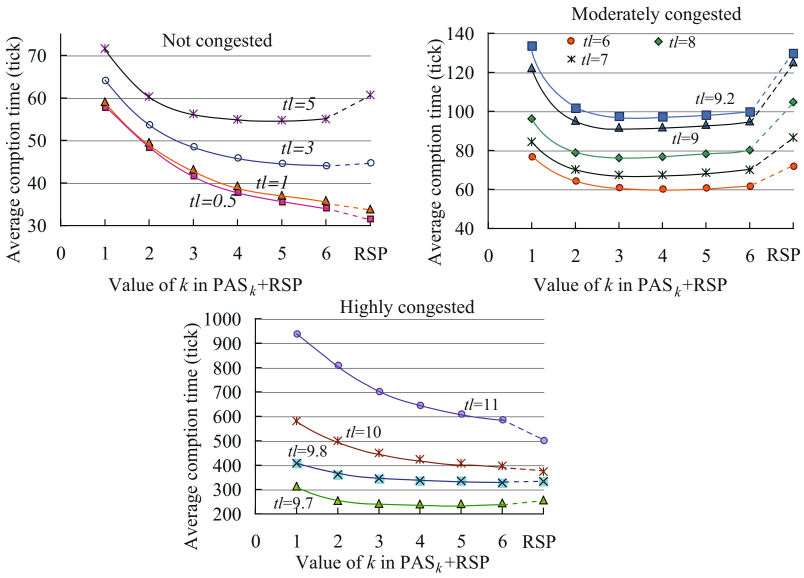


Fig. 1. Completion times under PAS$_k$+RSP(20)

3 Fluctuation in Award Selection

3.1 Previous Results

We first describe an announcement policy under which manager m_j announces tasks to only n contractors randomly selected from K_{m_j} to reduce the number of messages in CNP, where n, a positive integer, is called the *announcement number* and indicates the number of announcements. This *random selection policy* is denoted as RSP(n). This policy requires neither prior knowledge nor learning about the contractors, but tasks may sometimes not be announced to capable contractors.

We previously examined [9] how overall efficiency varies for n ranging from 5 to 50 and $0.1 \leq tl \leq 11$. We found that our expectation that a smaller n results in inefficiency in the MMAS because tasks may not be announced to capable agents applies only when the task load is extremely low. Because managers send more task announcement messages under RSP(n) for a larger n, we can predict task concentration in a few good contractors in busier environments, thus making the MMAS inefficient. We found that this phenomenon can be observed much earlier (even when the task load is low) than we expected, however.

We also tested a learning-based audience restriction policy [9] in which each manager learns which contractors are more capable by observing completion times. While this restriction policy did not lead to better performance as expected, we did find that a small degree of fluctuation in the award phase considerably improved the overall performance and reliability.

To understand the effect of fluctuation in the award phase more clearly, we investigated how the degree of fluctuations affects the overall performance[10]. After a task announcement, manager m_j receives bids from a number of contractors, $\{c_1, \ldots, c_p\}$. We denote the bid value from contractor c_i as $b(c_i)$; m_j selects an awardee, c_i, in accordance with the following probability:

$$Pr(c_i) = \frac{1/b(c_i)^k}{\sum_{l=1}^{p} 1/b(c_l)^k} \tag{1}$$

Note that smaller bid values are better. This *probabilistic award selection* control in the award selection is denoted as PAS$_k$. The policy combined RCP(n) with PAS$_k$ is denoted by PAS$_k$+RSP(n). The larger the k, the smaller the degree of fluctuation; and PAS$_0$ and PAS$_\infty$ correspond to ‘random selection’ and ’no randomness’, respectively; so PAS$_\infty$+RSP(n) is identical to RSP(n). Variable k is called a *fluctuation factor*, hereafter.

Figures 1 (a) to (c) show how overall performance varies under policies RSP and PAS$_k$+RSP; k ranges from 1 to 6 and the announcement number, n, is fixed at 20. Note that we show graphs for only $n = 20$ because the overall performance in this case is generally better than that in cases where fluctuation is introduced [9]. We set n to 20 for most of the experiments discussed in this paper, and the announcement number is often omitted if $n = 20$.

These figures illustrate that the RSP policy only results in better performance than PAS$_k$+RSP when the task load is less than 3 (not so busy) or more than 10 (extremely busy, i.e., over the theoretical limits of the entire MAS). In other situations where $3 \leq tl < 10$, some degree of fluctuation can result in much better performance, but the k value leading to the best performance depends on the task load. For example, when tl is close to three, a larger k is better, but when tl is greater than six, the value of k that expresses the best performance gradually approaches 3. However, if tl is larger than nine, the best k value swiftly approaches 6. This analysis suggests that the award selection policy must be sensitive to the task load of the MMAS.

3.2 Performance with Variable Task Load

Although Fig. 1 shows performance only when tl does not vary, the task load usually varies in real-world applications. We have now examined how the overall performance changes when the task load varies over time. The curves labeled “PAS$_3$+RSP(20)”

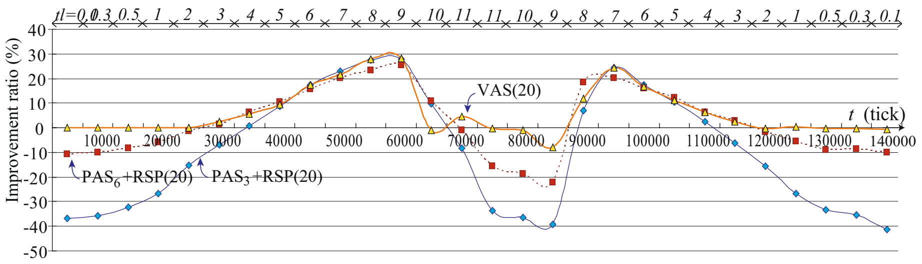


Fig. 2. Ratio of completion times under PAS$_k$+RSP(20) and VAS(20)

and “PAS$_6$+RSP(20)” in Fig. 2 show the *improvement ratio* (%) with PAS$_3$+RSP and PAS$_6$+RSP with respect to RSP; that is,

$$\frac{\wp(\mathrm{RSP}(n)) - \wp(\mathrm{PAS}_k + \mathrm{RSP}(n))}{\wp(\mathrm{RSP}(n))} * 100,$$

where $\wp(p)$ indicates the overall performance when policy p is used. In this experiment, tl started at 0.1 and gradually increased to 11 for 5000 ticks and then returned to 0.1. The improvement ratios are plotted every 5000 ticks. The values of tl are also shown in the figure.

Figure 2 not only clarifies the results of our previous experiments described in Section 3.1 but also suggests that, in the awarding phase, selecting the policy flexibly on the basis of the task load can improve the overall efficiency by as much as 30%. To evaluate the flexible control, we introduce *variable fluctuation control* into policy PAS$_k$+RSP(n) in which fluctuation factor is adaptively selected using the following *fluctuation control rule* (FCR):

$$\begin{array}{lll} k = \infty \text{ (i.e., RSP)} & \text{if } tl < 3 \text{ or } tl > 10, & \\ k = 6 & \text{if } 3 \le tl \le 5 \text{ or } 9 < tl \le 10, & \text{(R1)} \\ k = 3 & \text{if } 5 < tl \le 9. & \end{array}$$

This policy with FCR (R1), is called *variable awardee selection policy* and denoted by VAS(n). This FCR is induced from the experimental results shown in Fig. 1.

The overall performance under VAS(20) is also shown in Fig. 2. It indicates that, in general, VAS provides better performance than other policies using a fixed degree of fluctuation. The only exception appears when tl exceeds 9, that is, at around $t = 65000$, where the performance under VAS is lower than those under PAS$_k$+RSP ($k = 3, 6$). This is due to a small delay in shifting from the busy state to the unbusy state. Figure 2 shows that the system became overloaded after some tasks were stored in the queues by multiple awards; that is, the system got a little behind when the task load was beyond its theoretical upper limit. We can avoid this degradation in performance by delaying the switching from PAS$_3$+RSP to PAS$_6$+RSP and RSP, by modifying FCR (R1). However, this delayed switch resulted in another degradation at around $t = 85000$. This means that the control when the system is extremely busy is quite delicate and difficult.

4 Fluctuation Control Based on Estimation

4.1 Use of Queue Length

The major drawback of VAS is that it requires knowing the state of the system's task load, which is global information and usually unavailable in an open system like the Internet. To overcome this problem, we propose estimating the system's state from the bid values and supplemental information that is usually available from contractors.

With the proposed variable control, in the announcement phase, managers request the current queue length of each contractor as well as the bid values in their bid messages. This request is included in the bid specifications [1]. The managers can then estimate the task load (from their local viewpoints) using the queue lengths. First, suppose that manager m announces the task to n contractors, $c_1, \ldots, c_n$, randomly selected from K_m. It then calculates

$$r = \sum_{1 \leq i \leq n} \frac{q(c_i)}{n}, \tag{2}$$

where $q(c_i)$ denotes the queue length received from c_i. Ratio r is the average queue length of the contractors to which m made the announcement; it thus indicates how many tasks are simultaneously awarded (that is, the number of multiple awards). Then m select an awardee under VAS(n) but its fluctuation factor is determined by the following FCR:

$$\begin{aligned} &k = \infty \text{ if } r \leq 0.05 \text{ or } r > 2, \\ &k = 6 \quad \text{if } 0.05 < r \leq 0.15 \text{ or } 1.2 < r \leq 2.0, \quad \text{(R2)} \\ &k = 3 \quad \text{if } 0.15 < r \leq 1.2. \end{aligned}$$

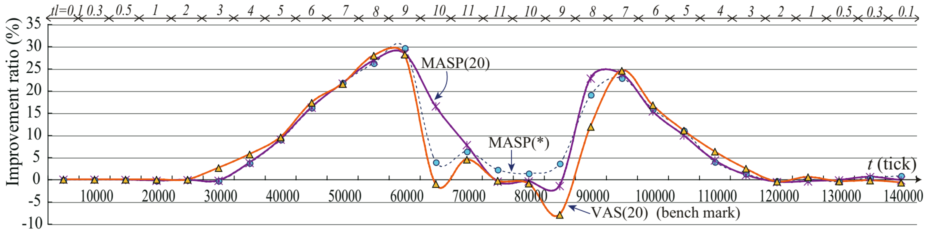


Fig. 3. Ratio of completion times under MASP(20) and MASP(*)

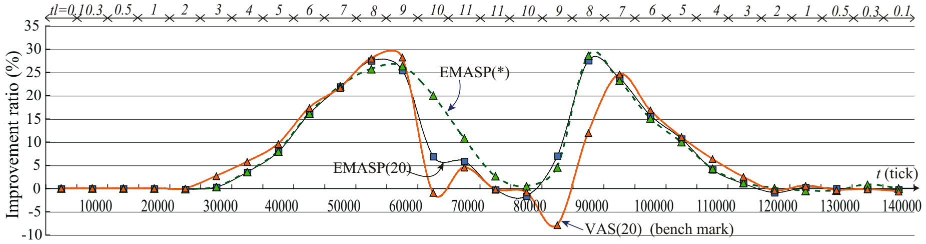


Fig. 4. Ratio of completion times under EMASP(20) and EMASP(*)

This policy with task-load estimation is called *multiple-award-number-based award selection policy* and denoted by MASP(n). Note that FCR (R2) is derived by modifying FCR (R1) and these threshold values dividing the award policies are derived from our prior experiments in which we investigated the relationship between the average queue length and the task load, tl. For example, when $tl = 9$, the average queue length reported with bid messages is approximately 1.2.

The curves in Fig. 2 indicate that the control when the system is busy ($tl \geq 9$) requires delicate attention, so we added another control for this situation. Our previous findings in [9] suggests that, by controlling announcement number n,we can improve overall performance. We thus added the control of announcement number into MAS: when $r > 2$, $n = 10$, and when $2 \geq t > 1.2$, $n = 5$, otherwise $n = 20$. This policy is called *MASP with variable announcement number* and is denoted by MASP(*). Note that there is another trade-off here; that is, reducing the number of announcements mitigates the concentration but degrades the accuracy of task-load estimation.

The overall performance characteristics under MASP(20) and MASP(*) are shown in Fig. 3. The performance under VAS(20) is also shown as the benchmark. The performance under MASP(20) was slightly better than that under VAS(20). We believe that this originates from small variations in the task load due to randomness; randomness does not mean uniformity. Thus, while estimation based on data from local contractors may not be accurate, it can reflect the local variations of the task load in a timely manner: these small variations, which can occur anywhere, significantly affect performance. When tl was 3 or 4 (t was around 30000 or 110000), VAS(20) had slightly better performance. This is because task-load estimation based on queue length is less sensitive when the system is not busy.

Figure 3 also shows that MASP(*) has better performance than the other policies when the task load is high. By tailoring the number of announcement when the MMAS is extremely busy, we can further improve the overall performance. Given these results, we believe that MASP(20) and MASP(*) are superior to VAS(20), because it is important to use all the capabilities of agents in MMAS when they are busy.

4.2 Estimation from Set of Bid Values

While the use of supplemental information, i.e., the queue length of each contractor added into the bid messages, is a promising idea, but we tried to estimate the queue length from only the bid values, that are information from the local contractors that is not fully utilized.

The idea is quite simple: managers learn the capabilities of their local contractors, which corresponds to how long it takes for them to complete a single task assigned, and then compare the bid values with the learned capabilities in order to estimate the queue lengths in individual contractors.

Suppose that manager m sends the l-th task announcement and receives bid messages from contractors $C_l = \{c_j\} \subset K_m$, where l is a positive integer. The ability of contractor c_i ($\in K_m$), as estimated from the l-th task announcement, is

$$a_l^{est}(c_i) = \begin{cases} \min\{b_l(c_i), a_{l-1}^{est}(c_i)\} & \text{if } c_i \in C_l \\ a_{l-1}^{est}(c_i) & \text{if } c_i \notin C_l, \end{cases} \tag{3}$$

where $b_l(c_i)$ is the bid value corresponding to the l-th task announcement from contractor c_i. Note that $a_0^{est}(c_i)$ is initially set a large number, such as 10000. Manager m calculates the values of $a_l^{est}(c_i)$ for $c_i \in K_m$. The derived value $a_l^{est}(c_i)$ converges with the minimum bid value from c_i, which also means the estimated completion time when c_i has no other unfinished awarded task.

The estimated c_i's queue length is

$$q_l^{est}(c_i) = \frac{b_l(c_i)}{a_l^{est}(c_i)} - 1,$$

which is used in Eq. (2) instead of $q(c_i)$. The modified MASP (with variable announcement number) using $q_l^{est}(c_i)$ is called *estimated multiple-award-number-based award selection policy* (*with variable announcement number*) and denoted by EMASP(n) (and EMASP(*)).

The performance under EMASP(20) and EMASP(*) are shown in Fig. 4. The performance under VAS(20) is again shown as the benchmark. Figures. 3 and 4 indicate that EMASP(20) and EMASP(*) can exhibit performance as high as those under MASP(20) and MASP(*) even though EMASP does not use the supplemental queue-length information from contractors.

An important issue to be addressed here is the accuracy of the $a^{est}(c_i)$ derived using Eq. (3). In the experiment shown in Fig. 4, the task load starts small: when a contractor is not busy, its bid value is the estimated completion time of only the announced task, which is the accurate data managers want. Conversely, if the system is busy, its bid values always include the execution time for other tasks in the queue in addition to that for the announced task. Therefore, the estimated $a^{est}(c_i)$ may be inaccurate and slowly converge to the actual $a(c_i)$.

To understand the effect of this slow convergence on efficiency when the task load is high, we set tl to 8 and investigated how the overall performance changed and how the ratio of the policies selected by managers under EMASP(20) changed over time. The results are presented in Fig. 5, where graph (a) shows the improvement ratios for EMASP(20), PAS$_6$+RSP and MASP(20) (which has almost identical features to PAS$_3$+RSP, which is thus omitted here) with respect to RSP(20), and graph (b) shows the changes in the ratio of policies selected by managers under EMASP(20).

Graph (a) in Fig. 5 shows that EMASP(20) becomes better than PAS$_6$+RSP(20) at around $t = 20000$ and approaches the efficiency under MASP(20) at around $t = 50000$. It always has better performance than RSP. Graph (b) shows that, at first, the ratio of mangers selecting PAS$_6$+RSP increased under EMASP(20). Because $a^{est}(c_i)$ gradually approached $a(c_i)$, they soon recognized that the local contractors had become a little busy, then they switched to PAS$_3$+RSP(20) over time. The two-dot chain line in Fig. 5 (b) indicates the ratio of PAS$_3$+PAS(20) managers selected under MASP(20) (almost constantly 96.1%). Other managers chose other policies because of the local small variations due to randomness. Note that initially all the managers chose RSP(20) since $a^{est}(c_i)$ always starts from c_i's first bid value.

For example, when $t = 20000$, managers announced 16 tasks on average since $tl =$ 8. This number is quite small considering the actual number of systems on the Internet. Of course, the task loads of actual systems always vary, and the systems are not always

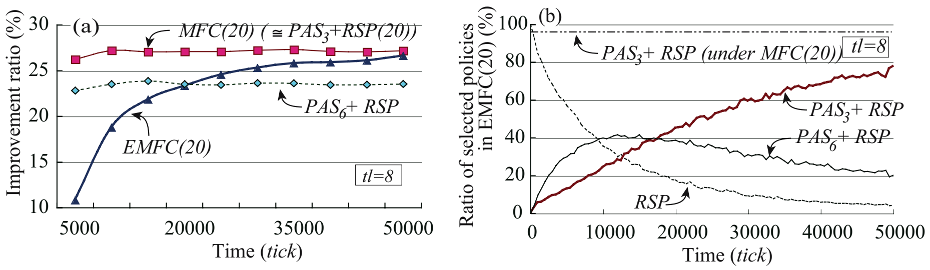


Fig. 5. Performance improvement and ratio of policies selected under EMASP(20)

Table 2. Ratios (%) of dropped tasks

Time range	65000–70000	70000–75000	75000–80000
RSP	6.85	10.28	3.65
PAS_3+RSP	4.86	9.58	3.75
PAS_6+RSP	6.91	10.86	5.42
VAS(20)	6.48	10.70	2.93
MASP(20)	6.75	10.98	2.94
MASP(*)	8.69	10.87	3.28
EMASP(20)	7.19	10.45	2.80
EMASP(*)	7.29	10.57	3.49

busy. When the task load is lower, managers can quickly learn the actual abilities of their local contractors. We can thus conclude that the proposed policy, EMASP, is well suited for CNP in MMAS.

4.3 Dropped Tasks

The better overall performance was not achieved at the expense of many dropped tasks. In our experiments, task drops were mainly observed only when t was in the range 65000 to 80000. The ratios between the observed numbers of dropped tasks and those of tasks generated in the simulation environment are shown in Table 2. Although the ratios under RSP_3+RSP(20) were slightly smaller and those under MASP(*) were slightly higher, no significant differences were found.

5 Discussion

The results of our experiments show that flexible control of fluctuation in award selection policy, which is controlled by k in Eq. 1, strongly affects the overall efficiency of an MMAS; a little capriciousness by the manager when making an award would significantly improve the overall performance. However, this suggests that rational decisions in award selection do not always lead to the best results. For truly rational decision-making, fluctuated decision has to be intentionally introduced to manager agents in situations where even their best decisions can negatively affect the efficiency of all agents in the MMAS. In addition to the fluctuation in an award phase, the overall performance is also strongly affected by announcement policy; there is an appropriate number of announcements corresponding to the task load. One of the key issues is how agents can identify situations in which they should be rational and those in which they should be a little capricious. In an open system like the Internet, however, agents must be truly autonomous: Therefore, they should recognize the situations and control their degrees of fluctuation in decision-making and the number of announcements on their own. The results of our experiments suggest that, because the task load is not uniform everywhere in real applications, it is better that managers autonomously identify their situations on the basis of their local viewpoints. The method proposed here provides one solution for this issue.

We have to discuss the effects of communications delay. If we carefully analyze the phenomena in our simulations, the main reason for multiple awards is communications delay. We can broaden the scopes of managers, but a wider scope results in longer delays and more frequent opportunities for multiple bids and awards, especially in busy situations.

We also note that the overall performance is affected by the topological structures of the physical (lower-layer) network. Recently, overlay networks reflecting application-level relationships among agents have received much attention. However, communication delay usually does not depend on the overlay network but on the physical structures of the Internet. Introducing network topologies among agents into our simulation is one of our next research topics.

It seems reasonable to add the queue length to the bid specifications since managers' scopes are restricted to the local contractors. Original information in bid messages may not reflect the performance and/or capability of each contractor. Even for such a case, the efficiency (or time-to-respond) is usually a matter of concern for Internet services and interactive systems. Thus, if the MMAS conditions cannot be estimated from the bid values, we think that additionally requesting the queue lengths is reasonable. Because bid messages are "word" coming from the local contractors, we believe that the use of bid messages is a good idea.

Finally, we must describe the importance of the system development methodology based on MAS simulation for large-scale applications. We obtained threshold numbers for switching the award-selection and announcement policies from simulations. One of the purposes of the simulations was to clarify the phenomenon, performance and operations of systems in an early state of development or when their actual testing and evaluation are impossible. The large-scale applications on the Internet are such systems; therefore, simulation-based performance tuning for MMAS should become more important. The importance of multi-agent-based simulation is now recognized and has been used for several applications such as design of pervasive computing applications [12], evaluation of high-performance cluster system [13], load-balancing in widely distributed systems [14] and explanation of phenomena occurring in markets [15]. Of course, we need further improvement in simulations so that they can accurately reflect real systems and, to this end, we should develop more reliable tools for simulating, for example, the Internet.

6 Conclusion

We have described a new flexible manager-side control policies for the contract net protocol that effectively uses the capabilities of all information agents in an MMAS. The basic ideas of our control policy are the use of bid values from local contractors to estimate the local state of the MMAS and that managers in the CNP autonomously and adaptively changes (1) the degree of fluctuation in the award policies and (2) the number of announcements, from their local perspectives. We showed experimentally that a control policy responsive to the local task load has better performance than a naive CNP and a CNP with inflexible control policies, even though our policy does not use global information.

The communication bottlenecks in broadband network are shifting from the communication links to the server nodes, so the control of load balancing among servers in large-scale and worldwide systems is becoming critical. A more sophisticated control that fully utilizes the potential capability of systems is required for future network applications, and our research is aimed at to this requirement.

Although we did not introduce a contractor control policy, it is clear that control policies on both sides are required for better performance; as a first step, we explored how overall performance can be improved with only manager-side control. We also assumed that (1) there are no dishonest contractors, (2) all managers have the same policy, and (3) there is no locality in the task load. Additionally, no task structures were assumed; this makes the interference among information agents induced by the CNP simpler. Nevertheless, we observed nontrivial characteristics of the overall performance of a large-scale MAS, in which a huge number of information agents interact with each other. This paper describes only the first proposal of control policy for the CNP, We intend to tackle these issues one by one, with the aim to develop the effective negotiation protocol for MMAS.

References

1. Smith, R.G.: The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. IEEE Transactions on Computers C-29(12), 1104–1113 (1980)
2. Sandholm, T.: An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In: Proceedings of the Eleventh National Conference on Artificial Intelligence, pp. 256–262 (1993)
3. Conry, S.E., Kuwabara, K., Lesser, V.R., Meyer, R.A.: Multistage Negotiation for Distributed Constraint Satisfaction. IEEE Transactions on Systems, Man and Cybernetics 21(6), 1462–1477 (1991)
4. Sandholm, T.: Automated Contracting in Distributed Manufacturing among Independent Companies. Intelligent Manufacturing 11(3), 273–286 (2000)
5. Yokoo, M., Sakurai, Y., Matsubara, S.: The Effect of False-name Bids in Combinatorial Auctions: New Fraud in Internet Auctions. Games and Economic Behavior 46(1), 174–188 (2004)
6. Weyns, D., Boucké, N., Holvoet, T.: Gradient Field-Based Task Assignment in an AGV Transportation System. In: Proceedings of 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), pp. 842–849 (2006)

7. Parunak, H.V.D.: Manufacturing experience with the contract net. In: Huhns, M. (ed.) Distributed Artificial Intelligence, pp. 285–310. Pitman Publishing, London and Morgan Kaufmann, San Mateo (1987)
8. Schillo, M., Kray, C., Fischer, K.: The Eager Bidder Problem: A Fundamental Problem of DAI and Selected Solutions. In: Proceedings of First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), pp. 599–606 (2002)
9. Sugawara, T., Hirotsu, T., Kurihara, S., Fukuda, K.: Performance Variation Due to Interference Among a Large Number of Self-Interested Agents. In: Proceedings of 2007 IEEE Congress on Evolutionary Computation, pp. 766–773 (2007)
10. Sugawara, T., Hirotsu, T., Kurihara, S., Fukuda, K.: Effects of Fluctuation in Manager-side Controls on Contract Net Protocol in Massively Multi-agent Systems. In: Proceedings of 2008 IEEE International Conference on Distributed Human-Machine Systems (2008)
11. Xu, L., Weigand, H.: The Evolution of the Contract Net Protocol. In: Wang, X.S., Yu, G., Lu, H. (eds.) WAIM 2001. LNCS, vol. 2118, pp. 257–264. Springer, Heidelberg (2001)
12. Ishida, T., Nakajima, Y., Murakami, Y., Nakanishi, H.: Augmented Experiment: Participatory Design with Multiagent Simulation. In: International Joint Conference on Artificial Intelligence (IJCAI 2007) (2007)
13. North, M.J., Hood, C.S.: A Multi-agent Systems Model of High Performance Computing Cluster Users. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 99–113. Springer, Heidelberg (2005)
14. Sugawara, T., Kurihara, S., Hirotsu, T., Fukuda, K., Sato, S., Akashi, O.: Total Performance by Local Agent Selection Strategies in Multi-Agent Systems. In: Proceedings of 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS2006), pp. 601–608. ACM, New York (2006)
15. Izumi, K., Yamashita, T., Kurumatani, K.: Analysis of Learning Types in an Artificial Market. In: Davidsson, P., Logan, B., Takadama, K. (eds.) MABS 2004. LNCS (LNAI), vol. 3415, pp. 145–158. Springer, Heidelberg (2005)

Filter Allocation Using Iterative ECNP

Jan Tožička, Štěpán Urban, Magdalena Prokopová,
and Michal Pěchouček

Gerstner Laboratory
Department of Cybernetics, Czech Technical University
Technická 2, Prague, 166 27, Czech Republic
{tozicka,urban,prokopova,pechouc}@labe.felk.cvut.cz

Abstract. Network devices can filter traffic in order to protect end-user computers against network worms and other threats. Since these devices have very limited memories and cannot deploy filters against every known worm, the traffic can be forwarded to other device during so called *filter delegation*. In this contribution we present two negotiation based algorithms looking for a good filter delegation solution. We formally describe this *filter allocation problem in a network* dealing with distribution of filters among agents so that several constraints are fulfilled and we extend this problem to fit a real world task. We show that both the basic problem and its extension are NP-complete. Both algorithms solving this problem are experimentally evaluated on a realistic network simulation.

1 Introduction

As data communication networks become more complex and its security more crucial then it ever was, new ways to monitor and protect them must be investigated. The software agent technology can be applied to many problems spaces from public and private to military networks on the battlefield.

Software agents seem to be a natural fit for this type of environment, they are small, robust, mobile, and have the ability to analyze and adapt to their local environment on the fly. This can also be very useful in networks with communication lags and dropouts, such as overloaded networks and wireless (Ad-Hoc) networks where the network structure can change.

Intrusion detection is an area where the agent technology can be applied. Agents that contain mobile and RC characteristics would provide a dynamic capability to discover and protect against emerging cyber threats. Knowledge sharing among agents would enable dynamic reconfiguration of the agent's capabilities and allow for distributed processing. The traditional method is to have an appliance, or application running on a server, then attack signatures would need to be put in by hand or downloaded and installed in order to add this new functionality. While the agents will still rely on similar approach and should be able to efficiently react to and contain known threats, they shall be able to use AI techniques to identify new threats similar to the known ones or emphasize irregular operations and cooperatively adapt using advanced negotiation techniques.

1.1 $\mathcal{A}$-net Network Simulation

In our work we focus on the domain of network security and reaction to intrusions. During our work we developed an agent simulation of computer network *A-net* (see Figure 1).

This agent-based network simulation provides us an easy way to deploy and test different network protection mechanisms. Each network device is represented by fully autonomous agent. The base layer of this simulation offers all common network devices, end user systems and the traffic between them. In this layer actual flow payload is not generated by the application itself, but rather one of the several hundreds of example packets is used instead. By having the large pool of example packets we can assume that this approximation is close enough to the reality. The data we are using were gathered from real-world network traffic outgoing and incoming to one of our host machines.

Additionally, we added viruses – worms that spread in the network and attack vulnerable systems – and DDoS attacks – group of bots generate huge amount of traffic targeted to one server. Using modeling of the network traffic, our intrusion detection system identifies these intrusions and creates descriptions of the malicious traffic that should be filtered out of the network, we refer to these as *filters.*

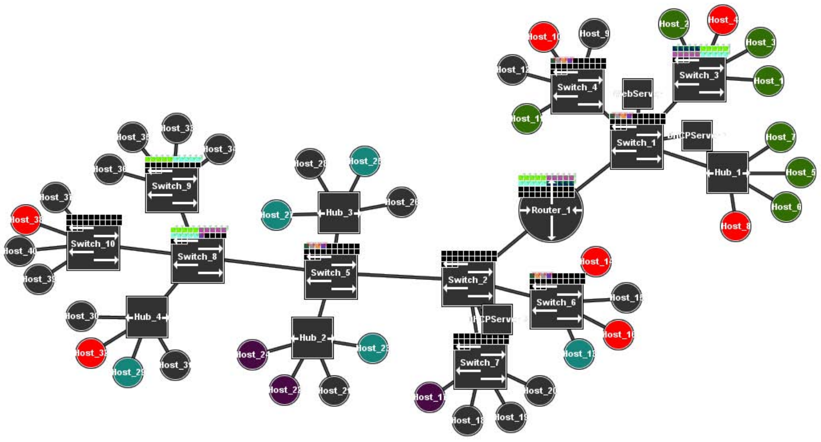


Fig. 1. $\mathcal{A}$**-net** network simulation

1.2 Network Flow Filters

Each network device, such as *router, switch, hub*, is equipped with a field-programmable gate array (FPGA) and a secondary memory. The gate array is very limited and it can contain only few filters. Nevertheless it is able to process a heavy traffic in real time (GBs per second) and filter out such a traffic that corresponds to the filters. On the other hand the secondary memory is big enough to contain specification of all worms ever created but it is very slow to be used by CPU to filter the traffic in real time.

Under these settings we try to protect all vulnerable hosts and to decrease the amount of malicious traffic in the network. In Figure 1 you can see that each network device (switches and router) have boxes which shows us deployed filters in our visualization. This box can be empty (no filter) or colored by color related to filtered traffic. Colored host means that this computer has been infected by a worm.

1.3 Filter Allocation Problem

In this article we focus on the distribution of *filters* in the network. The filters are distributed over a network so that they cover all paths between vulnerable hosts. This problem is formally described as *filter allocation problem in a network (FAPN)* and we show that it is NP-complete in the Section 4. To be able to solve real world problem we have extended the FAPN to FAPN-E which is *distributed*, *on-line*, *unbounded* and with *incomplete knowledge*. If an network device cannot deploy some of the filters other device covering the same path can deploy the filter with possibly worse utility function or the filter can be delegated to other netowrk device. In this article we present and compare two negotiation based algorithms looking for suitable filter delegation.

The rest of the paper is organized as follows. Firstly, we introduce work related to our problem in the Section 2. The Section 3 presents two algorithms searching for filter delegation. Firstly it presents ECNP based greedy algorithm in the Section 3.2 which is further extended to second algorithm described in the Section 3.3. We formally describe and analyze the complexity of FAPN problem and its extension FAPN-E in the Section 4. Both algorithms are experimentally compared in the Section 5 and we conclude this contribution in the Section 6.

2 Related Work

Balanced allocation of the monitoring process within the network as well as an efficient placement of the intrusion response processes need to be decided and reconfigured locally, in a peer-to-peer interaction among the network components. That is valid because there is a desire to limit centralized decision making processes and centralized collection of data in the network.

2.1 Service Oriented Architectures

Our domain is partially similar to service oriented architectures [3]. Service oriented architectures allow to create application distributed over different enterprizes. In some cases the services require local resources for their functionality while in other cases the services can be moved to other devices and used remotely. Our filters can be viewed as services. Nevertheless the relation of service consumer-provider is not well defined in the network domain but in the filter delegation where one agent explicitely expect other agent to do filtering on his behalf.

2.2 Task Allocation Problem

Distributed task allocation is a typical problem that is solved in its different variations in research communities (e.g. [10]).

The distributed task allocation algorithms are based on different auctioning approaches (e.g. English, Vickery, Dutch, Seal-bid, All-pay [6]), each having different properties in different environments. The most widely used approaches to distributed task allocation are based on the CNP (*contract-net-protocol*) [9] (based on single round Seal-bid auction), iterated CNP, its $OSCM$-CNP optimality improvements [7] and other combinatorial auctions. These techniques have been successfully used in a number of network oriented applications. [8] focuses on similar problem to our case, the agents have limited resources and preferences and thus they have to coordinate the task distribution. The tasks do not cost anything to be promised but their consume resources when they are performed. The situation with the filters is reverse: filter deployment consumes expensive memory but its execution is free.

The problem of distributing filters through out the network is similar to the problem of resource allocation and task distribution among autonomous agents defined in [1] where a set of agents share a common resource and an agreement is sought where all agents will be able to use this resource to fulfill their goals.

Similarly, in our approach each agent contributes to the negotiation task by several inputs:

– *filters* – Each agent has its list of prioritized filters it wants to deploy.
– *resources* – Each agent taking part in the negotiation also offers some resources for filtering, either to be shared or to be used by itself
– *capabilities* – Some agents can re-delegate their tasks to different agents, but others can not.
– *strategy* – The strategy can be defined by specifying how much traffic we allow to redirect and thus increase the network traffic

Based on the criteria for evaluating negotiation protocols presented in [1] we are looking for an algorithm that is (i) *distributed*, i.e. without any central point, (ii) *simple*, i.e. the negotiation consumes reasonable amount of resources. and (iii) symmetric, i.e. all agents should be treated in the same way.

[2] describes the similar problem of task distribution, where the problem scenario has following elements

– *manager* – Agent who owns the task is referred to as manager
– *contractors* – Agents willing to cooperate with manager are called contractors
– *task* – Each task has specified benefits and resources needed
– *resources* – Resources are distributed among agents

Agents are organized into social network according to who wants to cooperate with who and they cooperate only with their neighbors. The manager starts negotiation by offering the most efficient (ratio between benefits and required resources) task to its contractors and agents choose out of all tasks the most efficient one and make a bid.

As opposed to usual approach for task distribution in our domain we need different approach to the use of the resources. In the most common scenario resources can be used to satisfy only one task, on the other hand in our case since the resources are assigned to perform certain type of task, they can be used by large number of agents at the same time[1], but only for this purpose. Therefore some adjustments to existing algorithms are necessary.

3 Filter Allocation and Delegation in $\mathcal{A}$-net

Each agent desires to fulfill all necessary filtering tasks for the lowest cost possible but its resources are limited, therefore conflict of interests can arises. Thus in order to satisfy its needs agent has to reach an agreement with other agents. In [5], we have designed an algorithm that distributes a newly created filter through the network covering all vulnerable pairs of hosts and servers – i.e. satisfying the *completeness* condition (if such a distribution exists). Moreover this algorithm uses minimal amount of resources.

Once a new filter is introduced into the network the agent that received this filter sends it to its neighbor agents. Using the same technique the filter is distributed through the network. When the filter arrives to the host machine it replies whether it is vulnerable againts described threat. Agents collect these replies and based on them decide whether to deploy the filter and how to reply to the agent that informed it about this filter.

However this distribution algorithm is optimal only for the distribution of one filter, given several filters in a row the resulting distribution is not guaranteed to be optimal. Also, this algorithm does not use the filter delegation which could allow to deploy more filters in the network. Therefore we have implemented peer to peer negotiation that further improves filter distribution.

Since, in our network simulation, we allow for a simple delegation mechanism that allows to increase number of deployed filters in the network, a network device can ask another device to filter for it and then the respective incoming traffic is forwarded traffic to other device. These delegated filters are denoted as *filter-for*, or *FF*.

In the case a filter should be placed on a device not equipped with enough available resources the process of filter delegation starts to solve the situation. The idea of delegation is to find other devices with available resources where request for filter use can be forwarded.

Device can be chosen for filter delegation if

- the same filter is already deployed on the device, or
- the device has enough available resource to deploy the filter

The FAPN problem, formally described in the Section 4, is defined on an undirected graph $G = (\mathcal{A}, E)$, however we limit in our simulation to acyclic graphs only.

[1] Here, we consider resources needed for filter deployment only.

In this section we describe the requirements on such a solution in the Section 3.1 and then we present two solutions of filter allocation problem. Firstly, it is ECNP based greedy algorithm in the Section 3.2. We use this algorithm for evaluation of the second algorithm that uses more advanced negotiation based on an iterative modification of ECNP, described in the Section 3.3. Both algorithms are experimentally compared later in the Section 5.

3.1 Task Description

In this contribution we present and compare two algorithms finding the devices where the filters should be delegated to. ECNP protocol [4] addresses fast way how to distribute selected set of tasks. The improved iterative version of ECNP allows also to select good subset of tasks to be delegated.

When request for deploying new filter is obtained, the network device can reevaluate filters that are already deployed and according to priorities and statistics decide to remove one or more and place requested filter instead.

Apart from individual filter selection negotiation will be used also. Network devices can cooperate to find place(s) where to place filter while other devices will redirect their traffic to this place. The algorithm searching for optimal places can use following inputs and produces desired outputs:

Inputs:

– price for deploying filter
– price for redirecting traffic
  (The simple price for redirecting is given by number of nodes the traffic goes through. More elaborated measures take into account statistics – price for redirecting traffic for each filter can be multiplied by average usage of the filter.)
– priority of filter given at filter creation,
– statistics of filter usage/failures
  - number of flows filtered out / used
  - size of traffic filtered out / used
  - time since last successful filtering out / usage
– age of the filter

Outputs:

– what filters are deployed on devices
– what filters are delegated to other devices (traffic redirected)

Evaluation: Supposing we can order filters from best to worst using predefined metric (priority by itself or priority combined with statistics) we can divide the evaluation of the solution into two separate maximization/minimization tasks:

– firstly the task is maximizing number of pairs covered by filters where counting starts from the best filters
– on maximal coverage we need to minimize the increase in traffic in the network (this can be based on old statistics of filter usage and the distance how far the traffic is redirected)

By optimizing the above defined metrics the percentage of worm flows successfully filtered out should be the highest possible.

The algorithm delegating filters in the netowrk is also required to be:

– *distributed* – each agent is responsible for its netowork device and decides which filter to deploy locally
– *on-line* – filters are comming in sequence and agent need to improve current solution
– *stable* – the changes for new filter should be minimal since the cost of the reconfiguration of FPGA
– *unbounded* – there is no upper bound for the number of filters
– able to work with *incomplete knowledge* – some of the inputs are not known to the agents

3.2 Greedy Algorithm: ECNP

In the following paragraphs we describe the greedy algorithm that represents a lower bound solution of the problem.

The Algorithm 1 shows how we find new places where the filters are delegated. We are using ECNP, one of the extensions to CNP introduced in [4], where agents can bid only for parts of the offer, i.e. to choose only one or two filters out of the whole offer. Filters covered by winning bid are removed from the call-for-proposals (CfP) and the negotiation continues until all remaining filters can fit to the device itself. ECNP also introduces temporal grants and rejects, and thus the initiator of the negotiation can change its decision.

Algorithm 1. Filter delegation greedy algorithm – ECNP

```
while Enough resource to fit all filters are available do
    Send CfP for delegation of all filters that are not delegated yet to all agents.
    Wait for bids from all agents.
    Choose the best bid and confirm, reject others.
    Discard the filter locally and set delegation.
end
```

This solution is static and does not try to improve filter distribution once it is negotiated. It also delegates more filters than it is necessary. These imperfections are addressed by the improved algorithm described below.

3.3 Improved Algorithm: Iterative ECNP

In our scenarios we are facing the challenge of *incomplete knowledge* – agents can not estimate how much traffic the filter can filter out in advance. Anytime new filter is deployed on a device, the filter evaluation becomes available after some period of time when statistics are counted. Therefore we cannot take into account the amount of filtered traffic during the initial distribution and moreover this value can change in time, thus our solution needs to be periodically checked and adjusted.

The negotiation about filter delegation can be solved one filter by one, but better results are achieved when negotiating about set of filters. We propose an algorithm that finds the best set of filters to be delegated, minimizing the unavoidable increase in network traffic.

We modified ECNP protocol to build up our knowledge about how much will the traffic increase by using filter delegation. First the value of traffic increase is estimated for each filter and the set of filters with lowest value is chosen. We use ECNP to find possible delegation places for each filter and more accurate estimations, our implementation of ECNP is however modified and no proposal is accepted. Using these more exact estimation the set of filters for delegation is recounted and ECNP is started again. The details are described in the Algorithm 2.

Algorithm 2. Iterative ECNP

```
Choose initial set S of filters for forwarding.
Set price for forwarding to initial value for all filters.
while set S of filters to be forwarded changed do
    Start non-accepting ECNP algorithm.
    Reject proposals instead of accepting them.
    Recount eventual increase in traffic based on ECNP proposals.
    Choose new set S of filters for forwarding.
end
Start ECNP algorithm for final set S and deploy filters.
```

We cannot use simple estimation for each filter by itself, but all estimations has to be done over a set of filters. When estimating the value of increased traffic for a single filter the selected place for delegation is the closest agent with free resources, however this estimation can not be used for a set of filters, it is unlikely that the closest agent has enough free resources for all the filters. Therefore a set of filters is used for adjusting the estimations and the concrete distribution where to delegate is found once the set of filters for delegation is determined.

4 Formal Description of Filter Allocation Problem in Network

Let us now formally describe a *filter allocation problem in a network (FAPN)* in this section. Each agent a_i from the set of all agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ has limited resources r_i available to provide some of the filters. Let $\mathcal{F} = \{f_1, \ldots, f_m\}$ be a set of filters. Each filter $f_j \in \mathcal{F}$ is defined by a tuple $< u(f_j, a_1), \ldots, u(f_j, a_n), r(f_j) >$, where $u(f_j, a_i)$ is the utility for deploying the filter f_j by an agent a_i and $r(f_j)$ is a number of resources needed to deploy the filter.

In addition there are constraints where the filters are needed. These constraints are defined on a *network* represented by an undirected graph $G = (\mathcal{A}, E)$.

Each filter f_j determines a subset of paths in the G graph $\mathcal{P}_j \subseteq \text{Paths}(\mathcal{A})$, which needs to be *covered* by the filter f_j. Paths $\mathcal{P}_j$ represent paths between all pairs of vulnerable hosts in the case of worm filters or all path leading form bot net to the server they are attacking.

The *distribution* of filters between agents is defined by the function $\mathcal{D} : \mathcal{F} \mapsto \mathcal{A}$. A distribution is *valid* if it satisfies following properties:

correctness: Each agent $a_i \in \mathcal{A}$ does not deploy more filters then it has resources for: $\sum_{f_j \in \mathcal{F}:\mathcal{D}(f_j)=a_i} r(f_j) < r_i$

completeness: For each filter f_j all paths $\mathcal{P}_j$ are covered.

We suppose that such a valid distribution exists for each instance of FAPN.

Under these settings the task is to maximize the overall *utility* $\mathcal{U}$:

$$\mathcal{U} = \sum_{f_j \in \mathcal{F}} u(f_j, d(f_j))$$

Problems similar tasks to FAPN are often NP-complete (e.p. task allocation problem).

Theorem 1. *For an instance of a filter allocation problem in a network, as defined in the Section 4, and a real number k the problem to decide whether distribution $\mathcal{D}$ with utility higher than k exists is NP-complete.*

Proof. Firstly, let us show that FAPN is NP problem. Having an instance of the problem, real number k and a solution $\mathcal{D}$ we can check in a polynomial time whether it is valid distribution and whether its utility is greater than k.

We use knapsack problem (KSP) to show that FAPN is NP-hard, i.e. FAPN $\leq_p$ KSP. An instance of KSP contains n items, where an item i has value v_i and size f_i, and a size of the bag c. We look for a subset of items $S \subseteq \{1, \ldots, n\}$ that maximizes $\sum_{i \in S} v_i$ while fulfilling the constraint $\sum_{i \in S} s_i \leq c$. This instance can be transformed into FAPN with two agents $\{a_1, a_2\} : r_1 = c, r_2 = \sum_1^n s_i$ and n filters $\{f_1, \ldots, f_n\} : f_i =< \{u(f_i, a_1) = v_i, u(f_i, a_2) = 0\}, r(f_i) = s_i >$. A network connects both agents and they are present in all the sets $\mathcal{A}_i = \{a_1, a_2\}$.

A valid solution of this instance of FAPN problem can be easily transformed to a solution of original KSP problem. All items represented by filters deployed by the agent a_1 represent a solution S to KSP problem.

Variation of FAPN: FAPN-E. In this contribution we do not focus directly on presented FAPN problem but its variation will be considered instead. Let us change the FAPN problem in the following ways:

distribution – each agent is responsible for its filters

incomplete knowledge – non of the agents knows values of utility functions in advance but it can approximate the utilities of deployed filters.

on-line task – the whole set of filters is not known in advance but filters appear to the agents in sequence and the agents try to keep as good distribution as possible

unlimited size of $\mathcal{F}$ – the set of filters is increasing with the time and its size is unlimited. It means that after some time there is no valid distribution of FAPN task since the completeness property cannot be fulfilled. This case is described bellow in more detail.

The ever-growing set $\mathcal{F}$ will once harm the completeness property of each correct solution. Agents can deal with this problem in two ways. Firstly, an agent a_1 can delegate the filter f_j to another agent a_2 (that even does not have to be part of the path that need to be covered). This delegation consumes also a_1's r'_j resources. The utility of this delegated filter is the same as for the original filter $u'(f_j, a_2) = u(f_j, a_1)$ (if not considering price for delegation, e.g. traffic growth). Another possibility is that some of the paths remain uncovered in this case agents firstly try to cover as many paths as possible and afterwards they maximize the utility function.

Theorem 2. *The variation FAPN-E remains NP-complete.*

Proof. Following the proof of the FAPN NP-completeness it is obvious that the *distribution*, *incomplete knowledge* and *on-line task* cannot make the problem easier. The filter delegation nor the filter omission, that are used to deal with too large $\mathcal{F}$ sets, would not be used in the solution used in the original proof since they would not improve the solution of created task, e.g. in the case when $r'_j = r_j$. The filter omission will not be used since it would unnecessarily decrease the coverage.

Let us now describe our domain in the formalism presented above. The network we are protecting has a tree topology and the following items are essential for our problem:

agents: Each agent $a_i \in \mathcal{A}$ simulates a network device, such as *router, switch, hub or end user computer.* We assume that switches and routers are equipped with additional memory that can be further used.

resources:

filters: In our simulation we represent filters as *filters.* These filters can be loaded into additional memory of network devices and then filter the traffic going through the device. Each filter – filter f_j determines a set of agents it needs to *cover.* End user systems can have specified vulnerabilities and $\mathcal{A}_j$ defines a set of agents with the same vulnerability. We presume that worms can spread only between hosts with the same vulnerability.

network: All network devices are connected into the tree structure using the usual network topology.

utilities: The utility of each filter on particular device is how much it can *decrease malicious traffic* in the network, thus the sooner the traffic is filtered out the better and also the more flows go through given device the better.

5 Experiments

Several experiments were performed to show, how the Iterative ECNP (I-ECNP) can further improve the security of the system. Where I-ECNP is used more filters could be allocated into the network and more importantly their distribution

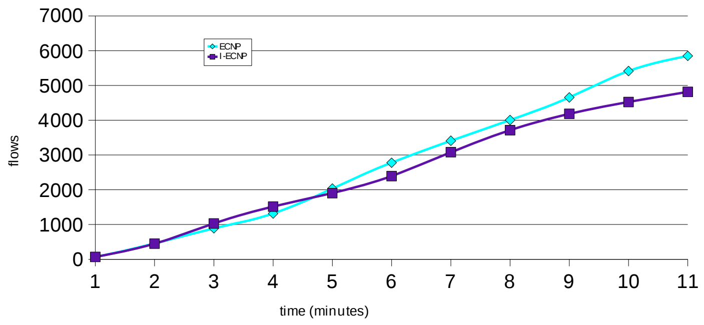


Fig. 2. Number of malicious flows reaching vulnerable hosts. This number is lower when Iterative ECNP technique is used.

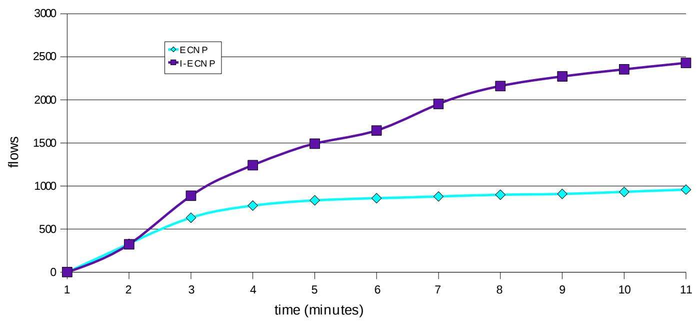


Fig. 3. Overall number of malicious flows that were filtered out by protection mechanism

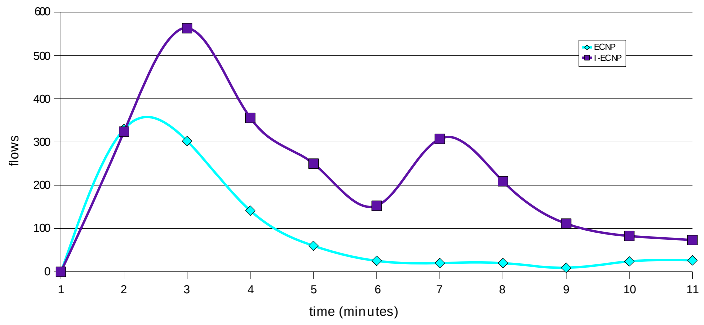


Fig. 4. Number of filtered out malicious flows per minute

in the network is adjusted for better performance therefore the system is better protected against malicious attacks. The following figures represent average from 3 test-runs using the original ECNP negotiation and 3 test-runs using the I-ECNP negotiation. All these experiments were evaluated on $\mathcal{A}$-**net** network simulation with one router and 5 subnets connected to this router. Each subnet contained 4 switchs, 15 hosts and one DHCP server.

Figure 2 shows comparison between number of malicious flows reaching vulnerable hosts with original negotiation used and with the new variant of Iterative ECNP. By improving the negotiation we achieved to lower the number of flows that reach vulnerable hosts and thus can infect these hosts.

Figure 3 and 4 show how the number of successfully filtered out worm flows increased. The amount of filtered out malicious flows is higher when the new Iterative ECNP technique is used. However after certain time, the network cannot accomodate more filters and the difference between those techniques is decreasing.

6 Conclusion

Even thought network devices with the capabilities assumed in this paper are available only in laboratories, we have investigated the algorithms for filter delegation. Both presented algorithms are based on the negotiation using ECNP communication protocol. First one tries to delegate the filters in a greedy way, while the other runs the ECNP negotiation repetitively to choose best subset of filters to be delegated. Both algorithms were evaluated in a realistic network simulation $\mathcal{A}$**-net** and the improvement of the iterated version of ECNP was shown. We also formally described solved problem and shown that it is NP-complete.

Considering the similarity of the problem solved in this contribution and service or task allocation problem, we believe that Iterative ECNP protocol can be usefull in these domains as well.

Acknowledgement

We gratefully acknowledge the support of the presented research by Army Research Laboratory project N62558-07-C-0007.

References

1. Cicortas, A., Iordan, V.: Multi-agent systems for resource allocation. In: 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, SACI 2005 (2005)
2. de Weerdt, M., Zhang, Y., Klos, T.: Distributed task allocation in social networks. In: Autonomous Agents and Multi-Agent Systems (AAMAS 2007). ACM Press, New York (2007)
3. Dijkman, R., et al.: The state of the art in service-oriented computing and design.
4. Fischer, K., Muller, J.P., Pischel, M., Schier, D.: A model for cooperative transportation scheduling. In: Proceedings of the First International Conference on Multiagent Systems, Menlo park, California, pp. 109–116. AAAI Press / MIT Press (June 1995)
5. Pěchouček, M., Tožička, J., Štěpán U., Prokopová, M.: Extending computational reflection in multiagent systems: towards autonomic computing. Technical report, The Gerstner Laboratory, Czech Technical University in Prague (2007)

6. Sandholm, T.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. In: Chapter Distributed Rational Decision Making, pp. 201–258. MIT Press, Cambridge (1999)
7. Sandholm, T., Lesser, V.: Coalitions among computationally bounded agents. Artificial Intelligence 94(1-2), 99–137 (1997)
8. Shehory, O., Kraus, S.: Coalition formation among autonomous agents: Strategies and complexity. In: Castelfranchi, C., Muller, J.P. (eds.) MAAMAW 1993. LNCS (LNAI), vol. 957, pp. 57–72. Springer, Heidelberg (1995)
9. Smith, R.G.: The contract net protocol: High level communication and control in a distributed problem solver. IEEE Transactions on Computers C-29(12), 1104–1113 (1980)
10. Walsh, W.E., Wellman, M.P.: A market protocol for distributed task allocation. In: Third International Conference on Multiagent Systems, Paris (1998)

On the Use of Symbolic Data Analysis to Model Communication Environments

Flavien Balbo and Julien Saunier

LAMSADE, Université Paris-Dauphine
Place du Maréchal de Lattre de Tassigny, Paris Cedex 16
{balbo,saunier}@lamsade.dauphine.fr

Abstract. Recent research on multi-party communications shows how multi-agent communications can take advantage of the complexity of the human communication process. The salient point is the very nature of the communication channels which enable humans to focus their attention on ambient communications, as well as to direct their own communications. For multi-agent systems, the difficulty is the routing of messages according to both the needs of the sender and the needs of the (potential) recipients . This difficulty is compounded by the necessity of taking into account the context of this communication. This article proposes an architecture for the Environment as Active Support for Interaction model (EASI) which is based on a classification data model and supports multi-party communication. Our proposition has been implemented and the functional description of the environment is given.

1 Introduction

Recent research on multi-party communications (MPC) [2,7,14,20,23] shows how multi-agent communications can take advantage of the complexity of the human communication process. In particular, the agents can have opportunistic behavior [12], can monitor the system [9], can have a support for information propagation [4]. The main issue in supporting MPC is to take into account dyadic interaction (one to one), group interaction (one to many) and overhearing (many to one/many) within the same interaction process. In MPC the sender viewpoint is not enough because it does not know all the agents that might be interested in its message. For example, an agent can listen to messages without the agreement/knowledge of the sender through overhearing [20]. Of course, the first MPC issue is the support of MPC itself, the second being the integration of context information, since part of the interaction is contextual and depends on the state of the environment [26]. For a recipient, the usefulness of a message may depend on the context of the sender, *e.g.* its location [11], the context of the message, *e.g.* its theme [4], and the context of the recipient itself, *e.g.* its availability.

These challenges are related to how the recipients are chosen. MPC requires knowledge of the needs of both the sender and the recipients [23]. In [20] and [23], the authors insist that the choice of recipients and the transmission of messages should be done by the environment. The environment is considered to be a first-class abstraction that embodies part of the responsibilities of the multi-agent system [27]. The use of context information requires a mechanism to access it and maintain it. Here, the environment can act as a context server [3]. This paper presents an architecture for the Environment as Active Support for Interaction (EASI) model which is based on a classification data model and supports MPC.

This paper is organized as follows. Section 2 describes the solutions to support interaction in MAS according to the MPC problematic viewpoint. Section 3 presents the adaptation of a classification data model for interaction purposes. Section 4 gives a functional description of the environment. Section 5 compares our proposal with related work; section 6 concludes.

2 Interaction Support

In a message exchange two sub-problems exist, depending on the viewpoint of the agents. From the senders viewpoint, it is a connection problem (CP): which agents are related to my message? The problem is to map the senders needs (information, capabilities[6], resources, ...) to the address of the related agents. From the recipients viewpoint, it is a data extraction problem (DEP): which messages are related to me? The problem is to map the recipients needs to the content of the messages. We classify the solutions to support interaction in three categories: 1) the interaction is dyadic, the solutions are based on direct communication between the agents; 2) the interaction is mediated, the solutions are based on message exchanges through a data space; 3) the interaction is "*organized*", the solutions are based on the use of a specific mechanism.

2.1 The Solutions Based on Dyadic Interaction

In an open and heterogeneous MAS, middle-agents are commonly used to look for an agent. The principle is to record in these specialized agents the information needed to look for the recipient with information about agent capabilities for the most usual. When an agent looks for an agent with a specific capability, it sends a message to the middle-agent and receives the list of potential recipients of its request. In the FIPA abstract platform architecture, this principle has been reused for the white or yellow pages directory services. The advantage in using a middle-agent is that it can support other services such as ensuring the anonymity of the agents. Depending on the services, a middle-agent taxonomy has been introduced in [29]. The problem created by centralizing the information is offset by the possibility of having several middle-agents. This solution implies common knowledge about these intermediaries and data updating has to be taken into account. Hence the success of this approach where information about agent capabilities means that not much data updating is required. Another frequently used solution in open and heterogeneous MAS is the Contract Net Protocol (CNP) [6]. The initiator broadcasts its request with the criteria that the agents have to satisfy to be selected for the following interaction process. If they want, recipients respond with their auto evaluation on the criteria. The success of this protocol relies partially on the fact that both sender and recipients are involved in the selection process [22]. Moreover, since the criteria are chosen according to the needs of the sender and independently of the protocol, the CNP has easily been adapted for several applications. The constraint is that it needs a broadcast at the beginning of the protocol. All these solutions are adapted to solve CP but they do not take into account DEP.

2.2 The Solutions Based on a Data Space

These solutions require the sender to put its message in a data space where the recipients read their messages. This data space can be external or can be a component of the MAS environment. In the first category, based on the LINDA model, there are the tuple spaces that are a continuation of the blackboard architecture. The aim of this approach is to avoid the space and time synchronization problems related to point-to-point communication. In this approach, the messages are tuples, and templates are used to look for them. The matching is done by comparing the message tuples and the template tuples according to the position and type of data used to describe the messages. TuCSoN has been developed using this model; it enables the tuple space to be programmed in order to specialize its reaction to the events, the result being what is called a Tuple center [18]. Using the same model, there is also LIME [17], which is a tuple space model for mobile agents.

Solutions based on the use of the environment extend the principle of a shared data space to take into account an interaction support in which the agents evolve. [19,27] propose a functional architecture of the environment which integrates a layer to support the interaction between agents. Communication between agents mediated by the environment comes from the reactive agent community and is often based on the stigmergy principle. Extended to cognitive agents, this trace mechanism has been used in [21] and [26], where all agent interactions are traces that can be observed by other agents and give extra information in addition to that contained in the message. In [8] the agents communicate through a tuple space that is integrated into the environment. These solutions take the viewpoint of the recipients and cannot be used to solve the CP.

2.3 Solutions Based on a Specific Mechanism

These mechanisms are based on the use of meta-knowledge or of a network protocol. In closed MAS and/or when the agents evolve within a platform, the MAS organization gives information to address messages. For example, in the Madkit platform[1], which is based on the organization model *aalaadin*, the agents communicate according to their role or their group in an organization. The structure of the organization is recorded within what is called the kernel. When an agent sends a message to agents according to a role, it sends the message to the kernel which looks for the recipients and puts the message into each of their letter boxes. In Magique[2], the MAS is organized within a hierarchical network where each agent is located on a node and the intermediaries nodes contain the skills of the agents situated below in the hierarchy. An agent that is looking for a skill sends the message to its superior and the message follows the hierarchy until the skill has been found. In these two examples, the agents do not have to know the address of a middle-agent or a tuple-space, they just use a common structure, their organization. Because the messages are not addressed according to the agent identifier or address, this solution improves the robustness and the efficiency of the MAS but limits communication to organization-based criteria. These solutions take the viewpoint of the sender and are solutions for CP.

[1] www.madkit.org

[2] http://www2.lifl.fr/SMAC/projects/magique/

The mechanism can be based on a network protocol. In [4], the authors use broadcasts restricted to specialized channels. The messages are sent on a channel to which the recipients have subscribed. Each channel has an IP address and is described by a string (following a taxonomy) in an XML file located on a URL. When an agent has chosen its channel it waits for messages. The messages are sent using the multicast function of the UDP protocol. In SIENA [5], the authors proposed a content-based routing protocol. Routing is done by filters that are introduced by recipients according to notifications that have been made by a sender. A notification is an item of information about the information that the sender could send on the network. The problem is to find the subset of recipients for each message and algorithms can improve filter management efficiency [25]. These solutions take the viewpoint of the recipients and are not suitable for the CP.

This discussion about the principal solutions that have been proposed to support interaction between agents shows clearly that each one is suitable for just one of the sub-problems. In this paper, we propose the EASI model, which takes into account, together and separately, the needs of both senders and recipients. The advantage is to have a single model that works with several interaction models and can easily combine them in order to take into account multi-party communication.

3 Communication Environment

In order for messages to be routed successfully, the environment has to stock a large set of data related to the agents, the message and everything necessary to know the context of the communication. In EASI, when a message is sent the environment searches through these data to choose the recipients. It has to gather information about the needs of the sender, the needs of the potential overhearers, and the context. Hence, our model has to address three issues: (1) an efficient search for recipients, (2) an expressive data description model and (3) a straightforward applicability.

Concerning issue (1), the environment contains information on all the components of the MAS and our objective is to let the agents use this common knowledge to define their interaction needs. For example, if information about the relationship between agents is available, then an agent a could send a message to the agents that it has in common with another agent b; in addition, an agent c, a friend of a but not of b and that might be interested could overhear the message. The amount of data increases fast, but the environment has to rapidly find the recipients and deliver the messages. Therefore, the model has to enable the efficient description and organization of large data clusters. Issue (2) is that of data representation, which should be expressive and enable a unified management of the descriptions and of the needs of the agents. The last issue (3) concerns the use of the model: it should be independent of the implementation, but also quickly applicable in real applications thanks to existing technologies.

These issues have led us to base our model on Symbolic Data Analysis (SDA) [1] in order to formalize the environment. SDA relies on the logic concepts of intension and extension to describe and classify data clusters. Its formalization is expressive since it does not depend on the type of data (quantitative, categorical or multi-valued), the comparison operators are given by the designer and variables can be used. SDA is applicable because the cluster definitions can be translated with SQL and/or in first-order logic.

Ω	p_1	p_2	p_3	p_4	...	p_{np}
ω_1	"a1"	"Main Hall"	44	false	...	$p_{np}(\omega_1)$
ω_2	"a2"	"A209"			...	$p_{np}(\omega_2)$
ω_3	"a3"	"A209"	37	true	...	$p_{np}(\omega_3)$
ω_4	"a4"	"A209"	33	true	...	$p_{np}(\omega_4)$
....	...	...	...	...	...	...
ω_n	$p_1(\omega_n)$	$p_2(\omega_n)$	$p_3(\omega_n)$	$p_4(\omega_n)$	...	$p_{np}(\omega_n)$

Fig. 1. Symbolic data table: n individuals and p properties

3.1 Symbolic Data Modeling

Let us begin by introducing the basic SDA definitions. The real world is made up of n individuals $\omega \in \Omega$. Each individual has np properties, and p_j is a mapping from Ω to D_j which associates to each $\omega \in \Omega$ a value in the definition domain D_j of the j^{th} property and $D = (D_1, ..., D_{np})$. For instance, individual ω_1 has four properties p_j: its identifier (string), its location (in a set of building positions), its age (number) and its availability (boolean). The values associated to ω_1 are $p_1(\omega_1) = "a1"$, $p_2(\omega_1) = "MainHall"$, $p_3(\omega_1) = 44$ and $p_4(\omega_1) = false$.

Considering the values of all the properties $p_j,\ j = 1, ..., np$ for the i^{th} individual, $d_{\omega_i} \in \mathcal{D}$ is the description of ω_i in the model. In the previous example, the description d_{ω_i} of the individual ω_1 is $"a1"$, $"MainHall"$, 44, $false$.

In practice, the symbolic data related to a given set of individuals are represented in an $n \times np$ matrix. The columns are the properties and the rows are the individuals. Figure 1 contains a symbolic data table, where the first rows and columns have been filled out. Not all the properties make sense for all the individuals, for instance the *age* of a message.

In the real world, the individuals can be grouped together thanks to concepts. The concepts $C_k,\ k \in \mathbb{N}$ are intents, they describe descriptions which satisfy certain individuals. An example of a concept C_1 would be all the individuals below 40 situated in room A209.

The extent of a concept is the set of individuals which satisfy this intent. In the table, the extent of the concept "younger than 40, in room A209" is the set $\{\omega_3, \omega_4\}$. An assertion is a symbolic object that is a mapping $\Omega \rightarrow \{true, false\}$. Let $v = v_1, ..., v_{np}$ be the description of an individual or a concept i, with v_j data that may be quantitative, categorical or multi-valued. An assertion is defined as:

$as = [p_{j1} R_{j1} v_{j1}] \wedge ... \wedge [p_{jq} R_{jq} v_{jq}]$ for $1 <= j1, ..., jq <= np$, where R_j is a comparison operator between the property p_j and the value v_j. The set of symbolic objects is $\mathcal{S}$. For example, the assertion $as_1 = [p_2(\omega) = A209] \wedge [p_3(\omega) < 40]$ is the description of the concept C_1. A symbolic object is an intent description. Its extent that is $E(a) = \{\omega \in \Omega | a(\omega) = true\}$ contains all the individuals which satisfy the comparisons with the description values of the assertion. For example, the extent of a_1 is a class of individuals which contains ω_3 and ω_4.

An assertion is therefore a comparison between the description of an entity and given values. It defines a class of entity which contains all the entities that satisfy these comparisons. A concept is also "*what the user needs*" with an exact but unknown extent

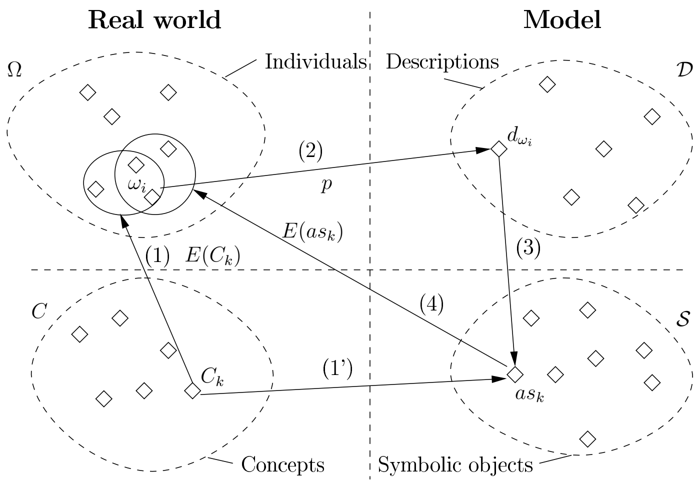


Fig. 2. Real world and model in Symbolic Data Analysis

in Ω, and a symbolic object is its formalization in the model with an imperfect but computable extent in Ω.

Figure 2 sums up the SDA definitions. The real world is made up of individuals and concepts. The concepts are intent descriptions of one or more individuals, for example "the agents available and situated in room A209". The extent of a concept (relation (1)) contains individuals $\omega_i \in \Omega$ which, taken as a whole, are a class. To each individual of the real world corresponds a description $d_{\omega_i} \in \mathcal{D}$ in the modeled world, thanks to the mapping p (relation (2)). A symbolic object is the formalization of a concept C (relation $(1')$). Symbolic objects $\mathcal{S}$ are intent descriptions using the descriptions of the individuals d_{ω_i} (relation (3)). The extent of a symbolic object contains the individuals of the real world whose description satisfies its description (relation (4)).

3.2 Communication Routing

This section shows how the Symbolic Data Analysis model is adapted to design a communication environment; more details about the EASI syntax can be found in [23]. In EASI, the environment contains the data for communication (Ω, $\mathcal{D}$) and information about how to manage it ($\mathcal{S}$). The concept is found at the MAS design level and not in the environment model. Some adaptation is direct: an individual is called an entity and Ω is the set of entities. An entity is the description of a component of the multi-agent system: agent, message, or object, the last one being a generic term to take into account anything which is neither an agent nor a message and that can be used to give information about the context of an interaction. For example, rooms are objects which can be described (size, capacity, facilities, etc.) and a message could be addressed according to this information: "message to the agents in the closest room". There is a distinction between the real component and its description in the EASI model. For example, an agent has its own process and knowledge, and a description of it is recorded in the environment. The set of properties $\mathcal{P}$ is the ontology of the communication environment, that is to say the information that can be used for communication purposes. $\mathcal{D}$ is the set of descriptions of the entities and completes the ontology with the values of the entities properties that are currently recorded in the environment.

Remember that a concept in ADS is an intent description of individuals. It represents what the agents need to identify in Ω, that is to say in all their common knowledge. A concept is reified by a symbolic object with an extent that is computed in Ω in order to directly identify the entities, or in $\mathcal{S}$ in order to find the symbolic objects that are related to the same need. There are three types of concept. The first type is related to the category of entities. A category of entities is a set of entities that is described by the same properties. This property set is called the *Pdescription* of the category, and the entities are clustered using the existence condition of required properties. Let C be a category and P_C its *Pdescription*; the assertion $cat_C = [P_C \subset P_\omega]$ is *true* for an entity $\omega \in \Omega$ if P_C is included in its own *Pdescription* (a $null$ value for a property expresses the absence of this property [23]. For example, let *FIPA* be the category of the FIPA messages that are recorded in the environment, let P_{FIPA} be the *Pdescription* of this category, $E(P_{FIPA}) = \{m \in \Omega | cat_{FIPA}(m) = true\}$ contains all the entities that have the description of a FIPA message (the FIPA language component is the properties in this case). This description level clusters the data in Ω according to their link with the communication needs. The advantage is that this link is independent of the property values and thus is independent of the update process.

The second type of concept is related to the communication needs of the agents, for example transmitting a message to "the agents available and situated in room A209". A filter is the reification in $\mathcal{S}$ of this need and gives the intent description of the constraints on the entities that are related to a connection. A filter $f \in \mathcal{F} \subset \mathcal{S}$ is a tuple $\langle f_a, f_m, [f_{co}], n_f, [priority_f], initiator_f \rangle$. The first three elements are assertions: f_a is the intent description of the constraints on the recipients, f_m is the intent description of the constraints on the messages and f_{co} is the intent description of the constraints on the context. The other elements are name, priority and initiator (agent that adds this filter to the environment) of the filter. Each assertion has a *Pdescription* too, for example P_{f_a} is the *Pdescription* of f_a and contains the properties that a recipient must have to be taken into account as a recipient.). The extent of a filter according to the property existence constraint in Ω contains the potential components of a connection that are gathered in the tuple $\langle E(P_{f_a}), E(P_{f_m}), E(P_{f_{co}}) \rangle$, with for example $E(P_{f_a}) = \{a \in \mathcal{A} | \forall p_i \in P_{f_{ag}}, p_i(a) \neq null\}$ with $\mathcal{A} \subset \Omega$ the agents set.

The last type of concept is related to the need to manage the relation between the entities and the symbolic objects. Basically, the idea is to match the $Pdescriptions$ of the entities to those of the assertions of the filters. A symbolic assertion is a mapping $\mathcal{S} \rightarrow \{true, false\}$. For a symbolic object, the symbolic assertion sa takes the value $true$ if the symbolic assertion is valid and false if it is not. For example, the relation between a message and its filters is designed as a concept: which filters f enable the message m to be received. Its reification is given by the symbolic assertion $sa_m(f) = [P_{f_m} \subset P_m]$ and its extent in $\mathcal{F}$ is $Channel_m = \{f \in \mathcal{F} | as_m(f) = true\}$, with P_{f_m} the *Pdescription* of the message related to the filter f. The result of this extent is a new symbolic object that we call SO and that has an extent in Ω. This extent is the union of the extent of the symbolic objects belonging to it and gives a solution to identify

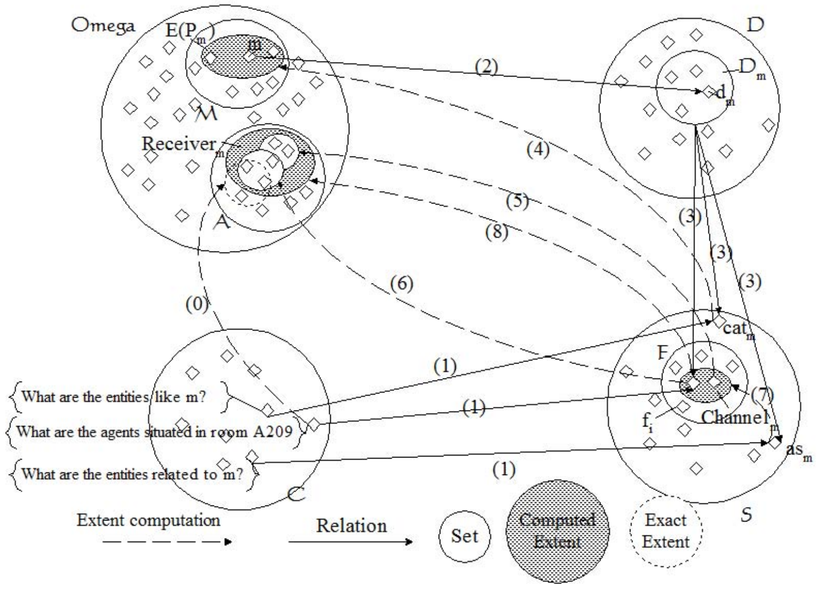


Fig. 3. Interaction component modeling

the relation between entities. For example, the extent of $Channel_m$ in Ω contains the entities that have in common the messages m; more precisely, this extent contains all its recipients ($recipient_m = \{a \in \mathcal{A} | \exists f \in Channel_m, a \in E(P_{f_a})\}$) and the whole context ($Context_m = \{C \subset \Omega | \exists f \in Channel_m, C \in E(p_{f_m})\}$).

In EASI, the filters dedicated to the management of the MAS belong to the environment. These filters are introduced either by a group of system agents, or by a mechanism which is internal to the environment. Thus, the filters are partitioned in two categories, depending on their initiator: $\mathcal{F} = \mathcal{F}_E \cup \mathcal{F}_A$, where $\mathcal{F}_E$ is the set of filters introduced by or on behalf of the environment and $\mathcal{F}_A$ is the set of filters introduced by the agents. In this way, the environment can add messages to the agents, i.e. they will receive messages that would not have been received otherwise.

Figure 3 shows the components of the EASI model and their relations. In Ω the entities are described with a value for each of their properties. This set is modified according to the update process of the MAS, such as D that gives a global overview of the entity values and therefore of the state of the MAS. $\mathcal{S}$ contains the tools to manage the recorded information. For example, a message is an entity m in Ω and d_m in $\mathcal{D}$ records its values (2). $\mathcal{D}_m \subset D$ contains all the values for each property that are related to the same category as m and enables the *Pdescription* P_m to be computed. According to this description and following the initial concepts (1),the assertion cat_{p_m}, the filters and the sa_m are computed (3). The extent (4) of the assertion (cat_m) related to the *Pdescription* P_m gives the entities that share the same properties. This cycle is used to find a subset of entities from one of their component or from a *Pdescription* coming from a user need (a concept). A filter is a subset of assertions and its extent is computed in Ω on the tuple that is composed of the extents (5-6) (only the extent for the agents

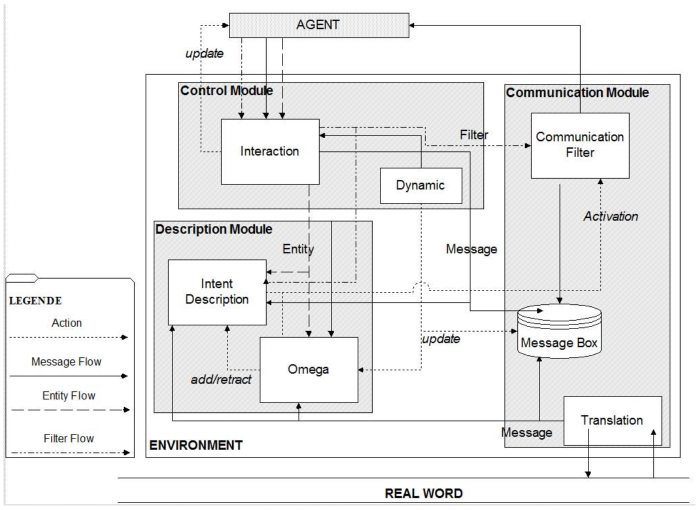


Fig. 4. An Environment Functional Description

are given in Fig. 3). The objective is to limit the processing to just the entities that are related to the interaction while remaining independent of the update process of the entities. The symbolic assertion sa_m and its extent $Channel_m \in \mathcal{S}$ are computed (7). The extent in Ω is the union of the extent of the filters (8). These data clusters can be computed *a priori* according to the description of the entities that are related to the interaction process in the MAS, which means that only the link between a new entities and these clusters has to be computed.

4 Functional Description of the Environment

This section provides a functional description of the environment (figure 4) that has been developed using the interaction model EASI. The proposal is based on the abstract functional description of the environment in [28]. The description is divided into two parts, firstly a description of the internal modules, then the common cases of interaction between agents and the environment.

4.1 Description of the Environment Modules

Our environment is composed of the modules related to the processing of the description of the MAS components (*Description Module (DM)*), to the processing of the communication (*Communication Module (CoM)*) and to the control of the communication (*Control Module (CM)*).

DM is composed of the sub-modules that manage the extent (*Omega* module) and the intent (*Intent Description (ID) module*) descriptions of the MAS components. *Omega* contains all the entities and enables modification operations (add/retract/modify).

ID contains the organization of the data as described in section 3.2. It contains the *Pdescription* of the entities, and for each filter the *Pdescription* of the assertions composing it and the symbolic objects that are related to the link between the entities and the symbolic objects. For example, for a filter f, *ID* contains P_{f_a}, P_{f_m},$P_{f_{co}}$ and for each of these sets their links with the other symbolic objects like $Channel_m$. *ID* maintains the link between the intent description and their extent in *Omega*, and the module containing the filters. When an entity *Pdescription* is added, if it is already recorded then the reference to the entity is added to all the structures the *Pdescription* is related to; in the other cases all the structures have to be tested. At this point the entity *Pdescription* are never deleted and this operation is under study. When the $Pdescriptions$ related to a filter are added, *ID* looks for SO where the filter has to be recorded; in the case of failure, new SO are created according to the *Pdescription* of the assertions.

CoM is composed of the *Message Box* sub-module where messages are stored, the *Communication Filters* sub-module where the filters are stored and triggered, and the *Translation* sub-module, which is an interface with the non EASI world. *Translation* stores received messages in the *Message Box* and ”translates” them using the EASI model. *Communication Filters* contains the filters of the agents ($\mathcal{F}_A$) and of the environment ($\mathcal{F}_E$). Filters are activated according to the modification in *ID*, which is then matched against the entity in *Omega*. When the matching is successful, the related descriptions are sent to the agent with the real message that is in *Message Box*.

CM contains the *Interaction* sub-module, which controls interaction between the agents and the environment, and *Dynamic*, which controls the update process. *Interaction* distributes the input of the agents in the environment (entity, messages, filter) and sends the agents the output of the environment (messages, update action). *Dynamic* maintains up-to-date *Omega*, it deletes messages that are too old (if a property related to time exist) or initiates an update of the agents properties. The objective is to give the MAS a global strategy in order to limit the cost of the update process. For instance, if agents have a property that gives their position, at least two strategies are possible: agents update their property each time they move or only when they are requested. A compromise has to be found between the number of errors and the cost of the update process.

4.2 Description of Common Cases

This section gives an example of MPC and concerns the dynamic management of resources in MAS. Each group of agents has a set of resources and, to simplify, each resource is unique and indivisible. An agent that needs a resource does as follows: it anticipates its needs and tries to know which agent in its group has its next resource or, if it cannot anticipate this, looks for the agent that has it. The owner of a resource accepts or refuses to give it to the requester. The objective is to avoid contacting all the agents of the same group each time one of them needs a resource.

Firstly, the agents record themselves in the environment. Each agent adds an entity to the environment that describes it (there is only the $group$ property in our example). *Interaction* adds the properties id (the value is unique and identifies the agent in the environment) and $address$ (the value is the address where the message will be sent). The entity is added to *Omega* and the *Pdescription* in *ID*. The latter module looks for symbolic objects that are related to this new *Pdescription*. For each of them that is modified the related filters are tested.

When an agent knows which resource it will have to use, it adds to the environment filters that match it against the description of the messages that are exchanged when an agent asks for a resource. There are two filters: the first (succeed) is related to when the resource owner agrees to give it and the second (fail) to when it refuses. *Interaction* adds each filter to *Communication Filters* and computes the *Pdescription*s (for example P_{fa}) that are added to *ID*. This module computes the extents in *Omega* and adds the result to *Communication Filters* to be triggered. If successful, the result is sent to the agents. The filter related to the success of the request informs the overhearer of the new owner of the resource and its behavior is not modified. The filter related to the failure of the request informs the overhearer of the present owner of the resource. In this case, the overhearer withdraws its filter related to the failure of the request. This filter will be added again if the resource owner responds favorably to a new request. The objective is to limit the number of useless messages. In the environment, the filter is deleted from *Communication Filter* and the related $Pdescriptions$ from *ID*.

When an agent needs to request a resource, it uses a filter that is already in the environment. This filter belongs to $\mathcal{F}_E$ and the content of this set is known to all the agents. Like in [28] the environment may or may not be distributed. In the first case, communication between environments is done through *Translation*; in the second case it is done directly with *Interaction*. In the two cases, the environment process is the same. The entity related to the message is added to *Omega* and its *Pdescription* is added to *Intent Description*. The choice of properties related to a message can be parameterized and additional properties like the date of the reception and the address of the recorded value in *Message Box* can be added. *ID* looks for the symbolic objects that are related to this new description. The set $Channel_m$ is added to *Communication Filters* to be triggered.

5 Discussion

This section compares our proposal with those presented in section 2, using three criteria. The first is the *expressiveness* of the proposals. A proposal is more expressive than another if it can take into account more complex constraints in the search for agents related to the interaction. The second criterion is the *completeness* of the proposal. A proposal is more complete than another if it can be used in more interaction models. The third criterion is the possibility of having *context aware* interaction. A proposal is more *context aware* than another if it can take into account more complex constraints on the context in the search for agents related to the interaction.

5.1 Expressiveness of the Delivery Mechanism

EASI is a model that aims to organize the data required to realize an interaction. For solutions based on dyadic interaction, it is necessary to compare our proposal with the matching process performed by the matchmaker or the sender of the messages. The matchmakers that come closest to our proposal are those that enable content-based routing [15,24]. In the case of solutions based on the CNP, the sender receives the evaluation of the agents on the requested properties and can also apply at least our selection process. For solutions based on a tuple space, it is necessary to compare matching on tuples and our solution. In tuple-spaces, the matching is limited to the type, position and value of the components of the tuples, while EASI enables the use of comparison operators, and the matching between entities. This is because in these works, the authors focused on the interaction model and not on the data model. In [8] the data model is not described, only its organization is: an input and an output matrix. The aim, for each item of data, is to find its link with agents. The senders are the lines of the matrix *outbox* and the recipients are the lines of the matrix *inbox*. As in EASI, set modeling gives a sufficient level of abstraction to manage the data and to use projections to identify the searched data. The operators that are used to do these projections are not included in the model and depend on the implementation phase.

In SIENA [5], a content-based routing model, filters are matched against single notifications (a tuple $<< type, name, value >^+>$) and patterns are matched against one or more notifications. The matching is done by comparing the value in the notification and the constants in the patterns. There are no variables that can be used to generalize the patterns. Moreover, in EASI part of the matching is done on the recipient and the context although only the data are evaluated in SIENA. Based on this criterion, this work comes closest to our proposal.

To sum up, in the majority of this research, the expressivity of the model is not taken into account and emphasis is put on other problems such as the interaction model in the tuple model or scalability in the delivery mechanism. This means that each of these solutions is complementary to the EASI model.

5.2 Completeness of the Delivery Mechanism

Using this criterion, the evaluation depends on the ability of a solution to enable: 1) a sender to find a recipient; 2) a recipient to choose its messages; 3) mutual-awareness. The first two points have been discussed in section 2 and we have shown that each of the solutions has been designed to take mainly into account the sender or the recipient. Mutual awareness is the possibility to receive messages that are sent to other agents. From the delivery mechanism viewpoint this means that the sender can choose its recipient and that another agent than the recipient can choose to receive the message. The delivery mechanism must also support both the search for recipients by sender and the search for messages by recipients. This is why the other solutions that take mutual awareness into account are mainly based on broadcast [13,16].

If mutual awareness is extended to the case where an agent can choose to receive messages coming from a subset of agents, then mutual awareness is a subcase of the delivery mechanism used to choose a message. In this case, solutions based on a shared data space and content-based routing solutions can be used if the data exchanged contains information about the sender. However, the advantage of anonymity found in these solutions is lost. Middle-agents mediate the search for receivers. Instead of delivering the messages according to both the senders and the receivers needs, the choice is restricted to a subpart of it. Furthermore, middle-agents are autonomous, while the environment is a supporting infrastructure [27].

The channeled multicast is a restricted broadcast and the recipients can be described as listeners of a subset of subjects, where a subject could be an agent. With the use of filters, EASI takes the needs of the sender and the recipients into account independently and simultaneously, according to the agent that has introduced the filter. If only the needs of the sender are taken into account (the filter is introduced by the sender of the message) then EASI is used as a delivery mechanism to choose the recipients. If only the needs of the recipients are taken into account (the filter is introduced by the recipients of the message), then if the message is not addressed EASI is used as a delivery mechanism to choose messages. If the message is addressed (filter introduced by sender) and heard (filter introduced by recipient) then EASI is used as a delivery mechanism to choose both recipients and messages and thus enables mutual awareness.

5.3 Expressiveness of Context Awareness in the Delivery Mechanism

The evaluation of each solution depends on if and how it can take the context into account in the search of the recipient or the message. This criterion is related to the expressiveness of the solution and to the availability of context-related information. Expressiveness is related to the expressiveness of the delivery mechanism and has already been discussed in 5.1. This criterion implies that the delivery mechanism has access to the state of the components of the MAS. Not all solutions have been designed to support context-awareness interaction: middle-agent, CNP, organization and channeled multicast. These solutions cannot take into account any information other than that related to the message itself. The solutions based on a shared data space or content-based routing have been designed to support context awareness. The only restriction is about the information available. In these solutions, information about agents is not taken into account and so the interaction cannot be conditioned by the observable state of the agents involved.

6 Conclusion

Designing a middleware to support MPC not only requires taking the needs of the agents related to communication into account but also evaluating these needs according to the context. The most common proposal is the use of a dedicated blackboard [7,10], which is a static medium for sharing messages. Thanks to the design and use of filters, EASI is an answer to these issues. The advantages of our proposal, based on the ADS clustering model, are connected to the different ways in which the information is managed. It can be done at different levels of abstraction, from an entity to a concept. The advantage is to limit the cost of the entity-updating process by the use of direct links (concepts) between the entities and the data clusters (the concept *extents*). These links are computable *a priori* and are dynamically changed if the MAS is open and new entity categories are added. Another advantage is to be able to obtain a description of the interaction facilities of the environment directly. This description is related to the ontology of the description of the entities with their existing value and to the filter description too. A filter (specially the environment filter) can be described as an interaction service: dyadic, broadcast, multicast, related to position, *etc*. Finally, the homogeneous description of the information related to entities and filters (their $Pdescriptions$) simplifies a processing that can be based on the same structure (under study) that is used to record them. Our proposal can be distributed with agents that are recorded in several environments or with environments that are federated. The environment nevertheless remains centralized and will be "really" distributed when the data model is distributed. Future work is related to the distribution of the data model, including the distribution of the description of the entities and filters.

References

1. Billard, L., Diday, E.: Symbolic Data Analysis: Conceptual Statistics and Data Mining (Wiley Series in Computational Statistics). John Wiley, Chichester (2007)
2. Branigan, H.: Perspectives on multi-party dialogue. Research on Language & Computation 4 (2-3), 153–177 (2006)
3. Bucur, O., Beaune, P., Boissier, O.: Steps towards making contextualized decisions: How to do what you can, with what you have, where you are. In: Roth-Berghofer, T., Schulz, S., Leake, D.B. (eds.) MRC 2005. LNCS (LNAI), vol. 3946, pp. 62–85. Springer, Heidelberg (2006)
4. Busetta, P., Donà, A., Nori, M.: Channeled multicast for group communications. In: AAMAS 2002: Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems, pp. 1280–1287. ACM Press, New York (2002)
5. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems 19(3), 332–383 (2001)
6. Davis, R., Smith, R.G.: Negotiation as a metaphor for distributed problem solving, 333–356 (1988)
7. Dignum, F., Vreeswijk, G.: Towards a testbed for multi-party dialogues. In: Workshop on Agent Communication Languages, pp. 212–230 (2003)
8. Gouach, A., Michel, F., Guiraud, Y.: Mic*: a deployment environment for autonomous agents. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 109–126. Springer, Heidelberg (2005)
9. Gutnik, G.: Monitoring large-scale multi-agent systems using overhearing. In: AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, p. 1377. ACM Press, New York (2005)
10. Huget, M.-P., Demazeau, Y.: First steps towards multi-party communication. In: van Eijk, R.M., Huget, M.-P., Dignum, F.P.M. (eds.) AC 2004. LNCS (LNAI), vol. 3396, pp. 65–75. Springer, Heidelberg (2005)
11. Julien, C., Roman, G.-C.: Supporting context-aware interaction in dynamic multi-agent systems. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 168–189. Springer, Heidelberg (2005)
12. Kamali, K., Fan, X., Yen, J.: Towards a theory for multiparty proactive communication in agent teams. International Journal of Cooperative Information Systems 16(2), 271–298 (2007)
13. Kaminka, G., Pynadath, C., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research 17, 83–135 (2002)
14. Kumar, S., Huber, M.J., McGee, D., Cohen, P.R., Levesque, H.J.: Semantics of agent communication languages for group interaction. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence, pp. 42–47. AAAI Press / The MIT Press (2000)
15. Kuokka, D., Harada, L.: On using kqml for matchmaking. In: ICMAS, pp. 239–245 (1995)
16. Legras, F., Tessier, C.: Lotto: Group formation by overhearing in large teams. In: Dignum, F.P.M. (ed.) ACL 2003. LNCS (LNAI), vol. 2922, pp. 254–270. Springer, Heidelberg (2004)

17. Murphy, A.L., Picco, G.P., Roman, G.-C.: Lime: A coordination model and middleware supporting mobility of hosts and agents. ACM Trans. Softw. Eng. Methodol. 15(3), 279–328 (2006)
18. Omicini, A., Zambonelli, F.: Tuple centres for the coordination of internet agents. In: SAC 1999: Proceedings of the 1999 ACM symposium on Applied computing, pp. 183–190. ACM Press, New York (1999)
19. Platon, E., Mamei, M., Sabouret, N., Honiden, S., Parunak, H.V.D.: Mechanisms for environments in multi-agent systems: Survey and opportunities. Autonomous Agents and Multi-Agent Systems 14(1), 31–47 (2007)
20. Platon, E., Sabouret, N., Honiden, S.: Tag interactions in multiagent systems: Environment support. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 106–123. Springer, Heidelberg (2007)
21. Ricci, A., Omicini, A., Viroli, M., Gardelli, L., Oliva, E.: Cognitive stigmergy: Towards a framework based on agents and artifacts. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2006. LNCS (LNAI), vol. 4389, pp. 124–140. Springer, Heidelberg (2007)
22. Sandholm, T.W.: An implementation of the contract net protocol based on marginal cost calculations. In: Proceedings of the 12th International Workshop on Distributed Artificial Intelligence, Hidden Valley, Pennsylvania, pp. 295–308 (1993)
23. Saunier, J., Balbo, F.: An environment to support multi-party communications in multi-agent systems. In: Burkhard, H.-D., Lindemann, G., Verbrugge, R., Varga, L.Z. (eds.) CEEMAS 2007. LNCS (LNAI), vol. 4696, pp. 52–61. Springer, Heidelberg (2007)
24. Skarmeas, N., Clark, K.L.: Content-based routing as the basis for intra-agent communication. In: Rao, A.S., Singh, M.P., Müller, J.P. (eds.) ATAL 1998. LNCS (LNAI), vol. 1555, pp. 345–362. Springer, Heidelberg (1999)
25. Tarkoma, S., Kangasharju, J.: Optimizing content-based routers: posets and forests. Distributed Computing 19(1), 62–77 (2006)
26. 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)
27. Weyns, D., Omicini, A., Odell, J.: Environment as a first class abstraction in multiagent systems. Autonomous Agents and Multi-Agent Systems 14(1), 5–30 (2007)
28. Weyns, D., Parunak, H.V.D., Michel, F., Holvoet, T., Ferber, J.: Environments for multiagent systems, state-of-the-art and research challenges. In: Weyns, D., Van Dyke Parunak, H., Michel, F. (eds.) E4MAS 2004. LNCS (LNAI), vol. 3374, pp. 2–52. Springer, Heidelberg (2005)
29. Wong, H.C., Sycara, K.P.: A taxonomy of middle-agents for the internet. In: ICMAS, pp. 465–466. IEEE Computer Society, Los Alamitos (2000)

Commitment-Based Multiagent Decision Making

Viji R. Avali and Michael N. Huhns

Department of Computer Science and Engineering,
University of South Carolina Columbia, SC 29208 USA

Abstract. In a cooperative system, multiple dynamic agents work together and share resources to achieve common goals, while simultaneously pursuing their individual goals. Interactions among the agents in such a cooperative system are critical to its successful behavior, and we believe that commitments are the proper abstraction to characterize the interactions. Commitments then become the basis for monitoring and controlling the system and tracking the progress towards its goals.

Commitments are binary relationships that bind two agents: a "debtor agent" that promises to provide a particular service for a "creditor agent". Their role is to represent agreements between the agents and prevent potential conflicts while the agents collaborate to achieve the system's common goals, which are imposed from outside. But the willingness to participate in achieving the goals comes from within the agent and that is why the beliefs, desires, and intentions of the agents are crucial in formalizing commitments. In this paper, we have formalized commitments in terms of the agents' internal states of mind—their beliefs, desires, and intentions. This formalization addresses what it means for a participating agent to promise or to satisfy a commitment. The formalization uses a branching-time computational tree logic framework with commitment definitions and operations to define a commitment-centric cooperative multiagent environment.

Keywords: Commitments, BDI, CTL*.

1 Introduction and Motivation

In a cooperative system, multiple dynamic entities work together and share their resources to achieve common goals, while simultaneously pursuing their individual goals. In real-world business environments, participants interact by exchanging goods and providing services for each other. In seeking and providing services, the participants form associations, make promises, commit to levels of functionality and quality, satisfy what they promised, and attempt to achieve their intended goals. We believe that in an environment where software agents are the participants, it is the binary relationship of *commitment* [1,10,11,5] that associates the agents with one another and represents multiagent interactions. Commitments can characterize—from an external viewpoint—not only the interactions between the agents, but also the overall multiagent system behavior.

Recent work on the concept of commitments has provided ways for an agent to evaluate a commitment and decide whether or not to promise it (as the debtor of the commitment) or accept it (as its creditor). However, current theories for commitments deal with only a single commitment and do not provide any help to an agent in relating or comparing several commitments. For example, if an agent has made two or more commitments, in which order should the agent work to satisfy them?

Our approach to this problem is to use the agent's beliefs, desires, and intentions to make decisions about commitments. The agent can then decide rationally when to accept, abandon, cancel, or devote resources to a commitment. The agent can also decide rationally in which order to satisfy its commitments. Moreover, a *commitment-driven decision theory* can be utilized to expressively model a cooperative multiagent environment. Development of this comprehensive theory is one of our research objectives.

Such a development should relate commitments to their effects on each of the participating agents' own *internal state of mind*. What does an autonomous agent believe when it creates a commitment? What does such an agent desire when it cancels its commitment? Many similar questions need to be addressed in order to develop a commitment-driven decision theory.

There has been a lot of work done on the belief, desire, and intention (BDI) architecture. Cohen and Levesque [2] explore the rational balance needed among beliefs, goals, actions, and intentions using a linear-time model. Rao and Georgeff [9] present a possible-worlds formalism for the BDI architecture using Computation Tree Logic (CTL). This work mentions that the BDI architecture could be extended to commitments by considereing them as part of multiagent scenarios.

There is also a rich literature on commitments. They are now well defined [12] and there is a formal representation for commitment operations [6]. Branching-time computational tree logic has been used to describe a commitment's typical structure [13], life-cycle, and various operations that are involved throughout its existence. However, most of the endeavors have focused on the external structure, properties, and verification of commitments. They do not explicitly formalize how commitments are understood by the participating agents themselves. These two areas (the BDI architecture and commitments) have been addressed separately and there has been little attempt to combine them (cf. [4]). Our work aims to integrate the two areas and formalize commitments in terms of the agent's beliefs, desires, and intentions in a CTL* framework, as indicated in Figure 1.

An agent's beliefs, desires and intentions define its *internal state of mind*. This paper formally defines commitments in terms of participating agents' beliefs, desires, and intentions. We use Rao and Georgeff's BDI framework [9] and Emerson's CTL framework [3], as well as earlier definitions for commitments [12] and operations on them [6].

The structure of this paper is as follows: Section 2 describes the major domain-independent types of commitments we have identified. Section 3 introduces a commitment-driven service-oriented multiagent environment and presents its underlying assumptions. Section 4 revisits the BDI_{CTL*} framework to describe agents in this environment. Section 5 introduces commitments and operations on them. Section 6 develops our formalization of commitments in a BDI_{CTL*}

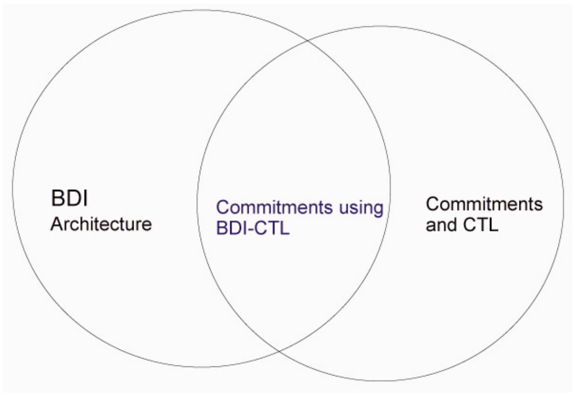


Fig. 1. Our results bridge BDI architectures, which are internal to agents, and commitments, which are public and involve the intentions among two or more agents. Our results are formalized using CTL*.

framework and provides definitions for all commitment operations. Section 7 presents an example of how our formalization can be used to explain, interpret, and model real-world multiagent systems. Lastly, Section 8 summarizes our formalization and presents future directions for this research.

2 Types of Commitments

Commitments associate one agent (the creditor) to another (the debtor) and are directed from the debtor to the creditor. They can be categorized into two basic types: discrete and continuous.

Discrete commitments have a lifetime, are created, remain active, and, at some point, cease to exist.
Example: agent Alice commits to pay $5 to agent Bob. The commitment ceases to exist when Alice pays Bob the money.

Continuous commitments are created and remain active indeterminately and until canceled. As an example of this type, a control system in a nuclear plant has a continuous commitment to maintain the coolant temperature within a desired range. Unlike discrete commitments, which are public, a continuous commitment might be visible only to the agent and is driven by the beliefs, desires, and intentions of the agent.

Each type of commitment can in turn be of two types. The first is the type that has a specific creditor, and these are what are typically thought of as commitments. The second type is when there is no specific creditor, and we call this an *obligation*. An obligation might be viewed as a commitment or a promise that one makes to oneself or to *society*. A society serves as an abstract creditor, which has been modeled as a Sphere of Commitment (SoCom) [12]. Examples

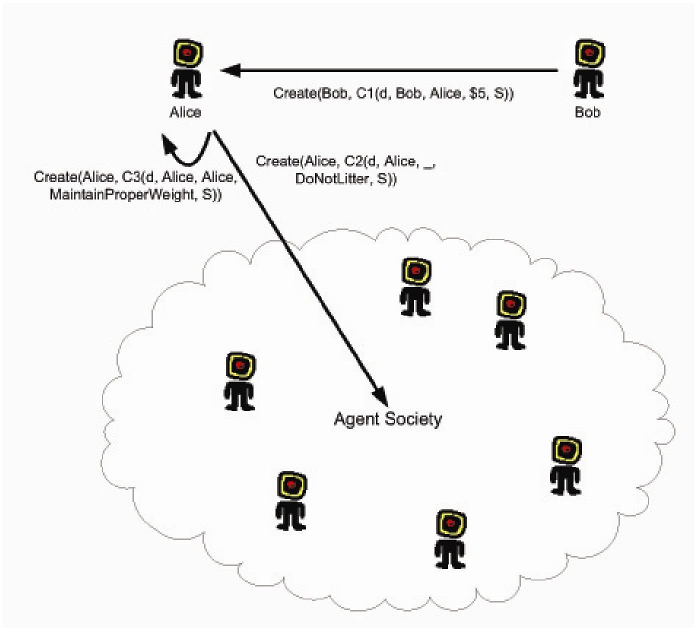

Alice
Bob
Create(Bob, C1(d, Bob, Alice, $5, S))
Create(Alice, C3(d, Alice, Alice, MaintainProperWeight, S))
Create(Alice, C2(d, Alice, _, DoNotLitter, S))
Agent Society

Fig. 2. The two major types of commitments are discrete ones between two agents and continuous ones between an agent and its society or itself

of continuous obligations are where one feels obligated to "honor your parents," "hold a door for the next person," "do not litter," and "protect the environment." Note that the potential actions involved in these might be positive (do something), negative (do not do something), or abstract (honor).

The two types of commitments are depicted in Figure 2. In this paper, we focus on formalizing discrete commitments.

3 A Commitment-Driven Multiagent System

Our formalization considers a multiagent environment that is partially observable, stochastic, sequential, and dynamic. The environment is cooperative and consists of two classes of participating agents: service providers and service seekers. Service providers and service seekers associate or bind with each other via the binary relationship of commitments. In addition to these participating agents, there is a class of *nonparticipating agents* in the environment that behave as impartial arbiters. The arbiters provide the context to a commitment relationship, as a SoCom. Every agent in the environment is autonomous; hence, at any point in time, a providing agent may choose to either abide by its commitment or stray from it. The arbiters can be used to capture a participating agent's behavior with regard to its commitments. Historical information about a participating agent's behavior can be utilized to measure its commitment adherence for future interactions, which is an area for further research.

Our formalization assumes that the participating agents have already identified each other and have already become part of a commitment relationship. How service seekers and service providers locate each other, how they identify compatible providers or seekers, how they interact or negotiate to form a binary commitment relationship, and what structure of communication and protocol they use are questions beyond the scope of our formalization.

It is further assumed that, in this commitment-driven cooperative environment, the partial view that an agent has is governed solely by the commitment relationships in which it participates. In other words, agents have knowledge of other agents with whom they are associated via commitment relationships. Furthermore, it is assumed that the knowledge about a commitment relationship is governed by commitment operations, i.e., an agent has knowledge about a commitment association only through operations that affect that commitment. For example, when a service-seeking agent and a service-providing agent participate in a commitment relationship, each will have knowledge of the other agent's commitment actions and each will have knowledge of when that commitment gets created, satisfied, canceled, etc. However, knowledge such as how that commitment is satisfied, why it was or was not satisfied, or why it was canceled is not available to the participating agents.

The typical environment for commitments is dynamic and nondeterministic, hence its temporal dimension is best represented as branching time. As the underlying temporal parameter moves forward, choices of actions by agents introduce branches, thus forming a tree. The following section describes in further detail this temporal structure as it relates to an agent's state of mind.

4 BDI in a Branching-Time CTL* Framework

In this section, we first restate Rao and Georgeff's BDI_{CTL} formalism, which is an extension of Emerson's Computation Tree Logic (CTL). In this formalism, the world is modeled with the help of an underlying branching temporal structure called a time tree, which has a single past and a branching time future, i.e., each moment on this infinite time tree may have many successor moments. It is assumed that along each path in this tree the corresponding timeline is isomorphic to $\mathbb{N}$. The maximal set of linearly ordered moments along a timeline makes a world and any point in a particular world is called a situation.

CTL operators are used to quantify over possible paths and states, and the temporal operators *A, E, X, U, F,* and *G* have their usual meanings (*A*: for all futures, *E*: for some futures, *X*: next, *U*: until, *F*: eventually, and *G*: always). BDI operators B, D, and I are used to represent the agent's internal state of mind.

4.1 Syntax and Semantics of BDI_{CTL*}

A particular point in a particular world is called a situation. A structure M with many such situations is a *Kripke* structure.

$$M = \langle S, R, B_a, D_a, I_a, L \rangle$$

where,

- S is a set of states.
- R is a binary relation $R \subseteq S \times S$.
- $L : S \rightarrow PowerSet(AtomicPropositions)$ is a labeling that associates with each state s an interpretation $L(s)$ of all atomic propositions at state s.

The relations B_a, D_a, and I_a map the agent's current situation to its belief, desire, and intention-accessible worlds. The structure M at a particular time point or moment m is denoted by M_m.

Assuming n agents, we define a set of admissible rules for States and Paths (true or false for states and paths) as follows:

State formulas:

(S1) Each atomic proposition is a state formula.
(S2) If α and β are state formulas, then so are $\alpha \wedge \beta$ and $\neg\alpha$.
(S3) If α is a path formula, then $E\alpha$ and $A\alpha$ are state formulas.
(S4) If α is a state formula,then $B_a(\alpha), D_a(\alpha), I_a(\alpha)$ are state formulas as well.

Path formulas:

(P1) Any state formula is also a path formula.
(P2) If α and β are path formulas, then so are $\alpha \wedge \beta$ and $\neg\alpha$.
(P3) If α is a path formula, then $X\alpha$ and $\alpha U \beta$ are path formulas.

A formula is interpreted with respect to a situation structure M. A *fullpath* x is an infinite sequence $s_0, s_1, s_2, ...$ of states, such that $\forall i (s_i, s_{i+1}) \in R$. A *suffix path* x^i is an infinite sequence $s_i, s_{i+1}, s_{i+2}, ...$ of states. We write $M \models_{m_0} p$ to mean that state formula p is true in structure M at moment m_0. We write $M \models_x p$ to mean that path formula p is true in structure M at fullpath x. $B_a(\alpha), D_a(\alpha)$, and $I_a(\alpha)$ are beliefs, desires and intentions of agent a about α.

(S1) $M \models_{m_0} p$ iff $p \in L(m_0)$,
(S2) $M \models_{m_0} p \wedge q$ iff $M \models_{m_0} p$ and $M \models_{m_0} q$,
$M \models_{m_0} \neg p$ iff not $(M \models_{m_0} p)$,
(S3) $M \models_{m_0} Ep$ iff $\exists$ fullpath $x = (m_0, m_1, m_2, ...)$ in M that $M \models_{m_0} p$,
$M \models_{m_0} Ap$ iff $\forall$ fullpath $x = (m_0, m_1, m_2, ...)$ in M that $M \models_{m_0} p$,
(S4) $M, m_0 \models B_a(\alpha)$ iff $\forall m_1 \in S$ and $m_0 R_1 m_1, M, m_1 \models \alpha$ where 'R_1' is the accessibility relation.
(S5) $M, m_0 \models D_a(\alpha)$ iff $\forall m_1 \in S$ and $m_0 R_2 m_1, M, m_1 \models \alpha$ where 'R_2' is the accessibility relation.
(S6) $M, m_0 \models I_a(\alpha)$ iff $\forall m_1 \in S$ and $m_0 R_3 m_1, M, m_1 \models \alpha$ where 'R_3' is the accessibility relation.

(P1) $M \models_x p$ iff $M \models_{s_0} p$;
(P2) $M \models_x p \wedge q$ iff $M \models_x p$ and $M \models_x q$,
$M \models_x \neg p$ iff not$(M \models_x p)$;
(P3) $M \models_x pUq$ iff $\exists i[M \models_{x^i} q$ and $\forall j(j < i$ implies, $M \models_{x^j} p)]$

5 Commitments

Now that we have described our multiagent environment and the state of mind of its participating agents, we define commitments and the operations that the agents can perform on them. For this purpose, we briefly revisit Singh and Huhns's formalism of commitments [12] and extend the commitment properties and operations defined therein.

Our formalism considers social commitments that are legal abstractions associating one agent with another. Earlier works have described another class of commitments that are personal or internal to an agent and do not bind two separate agents. However, such unitary internal commitments are not relevant to our cooperative environment, which is driven solely by binary relationships between the agents. Commitments are accessible publicly and they represent an interaction between two participating agents. For example, service level agreements, online purchases, and service contracts are all real-world instances of commitments.

As per the commitment formalism developed by Singh and Huhns[12], the following are three key properties of commitments:

1. **Multiagency:** Commitments associate one agent with another. The agent that promises or commits to satisfying a condition is called the debtor agent and the other agent that wants the condition to be satisfied by the debtor is called the creditor agent. Each commitment is directed from its debtor to its creditor.
2. **Scope:** Commitments have a well-defined scope, which gives context to the commitment. A scope can be directed by a separate third-party organization (Sphere of Commitment: SoCom).
3. **Manipulability:** Commitments are modifiable. They can be satisfied, breached, or canceled.

We extend these properties by defining two additional ones:

1. **Lifetime:** Commitments have a lifetime; they are created, they live (remain active), and at some point they cease to exist. Continuous commitments are beyond the scope of this paper and a subject of future research.
2. **Degree:** When active, commitments do not necessarily remain in one constant state; in real situations, at the time when people make commitments, they intend to fulfill them. But situations change and the priorities of commitments might thus change. This is captured by a *degree of commitment*. For a service-oriented environment, the degree of commitment changes with changing beliefs, desires, and intentions.

As an example, let us consider a travel agent who has a commitment to sell n tickets for airline A. If another airline (airline B) slashes their ticket prices and the customers want to buy those tickets, the travel agent reorders his commitments to satisfy his customers. Though he is still committed to A, his priority changes to selling airline B's tickets. Likewise, an individual agent can order any new commitment that he creates using a partial order. Anytime a change in his beliefs, desires, or intentions results in a change in preferences, he could reorder his commitments, thus mimicking the real life situation. The ordering method used would be dependent on the system.

Also, in the case of commitment cancelation or revocation, the commitment might not change from an active state to an inactive state instantaneously; instead, it might gradually decline in degree until it becomes inactive. This area is also a subject for future research.

5.1 Structure of Commitments

Commitments are represented by a predicate C and have the form $C(d, a, b, p, S, \delta)$, where

d: is a unique identifier,
***a*:** is the debtor agent,
***b*:** is the creditor agent,
***p*:** is the promise or the condition that the debtor will bring about,
***S*:** is the context, also known as the *sphere of commitment*, and
δ **:** is the degree of commitment.

For the sake of simplicity herein, we ignore δ.

5.2 Operations on Commitments

Our cooperative environment is commitment-driven and we assume the participating agents' knowledge is governed solely by commitment operations. Here we describe commitment operations as defined by [12,6], where commitments are treated as abstract data types that associate a debtor, creditor, promise, and context. The six fundamental commitment operations are

1. *Create (a, C(d, a, b, p, S))*
2. *Discharge(a, C(d, a, b, p, S))*
3. *Cancel(a, C(d, a, b, p, S))*
4. *Release(b, C(d, a, b, p, S))*
5. *Assign(b, c, C(d, a, b, p, S))*
6. *Delegate(a, c, C(d,a, b, p, S))*

We use predicates to describe whether the commitment C has been satisfied, canceled, breached, or still holds, and these predicates will be written as *satisfied(C), canceled(C), breached(C), and active(C)*, respectively [7].

6 Commitment Formalization in BDI+CTL*

In this section we present our formalization that represents a combination of BDI and commitments. Multiagent associations are bound by commitments and each agent's knowledge of those commitments is through commitment operations. Informally, a commitment between two agents comes about through interactions (and often negotiations) between the agents, so both agents are necessarily aware of and believe in the commitment.

Definition 6.1: Creating a Commitment,
Create(a, C(d,a,b,p,S))

1. For all paths, *Agent a* believes that from the next moment onwards commitment C will be active until it is either satisfied or breached or canceled.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_a((XG(active(C)))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
2. For all paths, *Agent a* believes that commitment C will eventually be satisfied.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_aF(satisfied(C))$
3. For all paths from the next moment onwards, *Agent a* intends the commitment C until it is either satisfied or breached or canceled.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AXG((I_a(C))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
4. For all paths, *Agent a* believes that from the next moment onwards *Agent b* desires commitment C until it is either satisfied or canceled.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_a((XG(D_b(C)))U$
   $(satisfied(C) \vee canceled(C)))$
5. For all paths, *Agent b* believes that from the next moment onwards commitment C will be active until it is either satisfied or breached or canceled.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_b((XG(active(C)))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
6. For all paths, *Agent b* believes that from the next moment onwards *Agent a* intends commitment C until it is either satisfied or breached or canceled.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_b((XG(I_a(C)))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
7. For all paths, *Agent b* believes that commitment C will eventually be satisfied.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AB_bF(satisfied(C))$
8. For all paths from the next moment onwards, *Agent b* desires commitment C until it becomes inactive.
   $M \models_m Create(a, C(d,a,b,p,S)) \Rightarrow AXG((D_b(C))U(\neg active(C)))$

Note that *agent b* can not intend C to be satisfied, because b does not have any control over C and cannot force it.

Definitions of other commitment operations can be written similarly.

Definition 6.2: Revoking a Commitment,
Cancel(a, C(d,a,b,p,S))

1. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg active(C)))$
2. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AB_aF(\neg satisfied(C))$
3. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AXG(\neg I_a(C))$
4. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AB_b(XG(\neg active(C)))$
5. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AB_bF(\neg satisfied(C))$
6. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AXG(\neg(D_b(C)))$
7. $M \models_m Cancel(a, C(d,a,b,p,S)) \Rightarrow AB_b(XG(\neg I_a(C)))$

Definition 6.3: Discharging a Commitment,
Discharge(a, C(d,a,b,p,S))

1. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_a(XG(satisfied(C)))$
2. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg active(C)))$
3. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AXG(\neg(I_a(C)))$
4. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg(D_b(C))))$
5. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_b(XG(satisfied(C)))$
6. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg active(C)))$
7. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AXG(\neg(D_b(C)))$
8. $M \models_m Discharge(a, C(d,a,b,p,S)) \Rightarrow AB_b(XG(\neg(I_a(C))))$

Definition 6.4: Releasing a Commitment,
Release(b, C(d,a,b,p,S))

1. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AXG(\neg(D_b(C)))$
2. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AB_b(XG(\neg active(C)))$
3. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg(I_a(C))))$
4. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg(D_b(C))))$
5. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg active(C)))$
6. $M \models_m Release(b, C(d,a,b,p,S)) \Rightarrow AXG(\neg(I_a(C)))$

Definition 6.5: Assigning a Commitment,
Assign(b, c, C(d,a,b,p,S))

1. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AXG(\neg(D_b(C)))$
2. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow B_bA(XG(\neg active(C)))$
3. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_b(XG(D_c(C)))$
4. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_b(XG(I_a(C)))$
5. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_c((XG(active(C)))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
6. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_c((XG(I_a(C)))U$
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
7. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_cF(satisfied(C))$
8. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AXG((D_c(C))U$
   $(satisfied(C) \vee canceled(C)))$
9. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_a(XG(D_c(C)))$
10. $M \models_m Assign(b,c,C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg D_b(C)))$

Definition 6.6: Delegating a Commitment,
Delegate(a, c, C(d,a,b,p,S))

1. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AXG(\neg I_a(C))$
2. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AB_a(XG(I_c(C)))$
3. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AB_a(XG(\neg active(C)))$
4. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AB_cF(satisfied(C))$
5. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AB_c((XG(active(C)))$
   U
   $(satisfied(C) \vee breached(C) \vee canceled(C)))$
6. $M \models_m Delegate(a,c,C(d,a,b,p,S)) \Rightarrow AXG((I_c(C))U$
   $(satisfied(C) \vee canceled(C)))$

7. $M \models_m Delegate(a, c, C(d, a, b, p, S)) \Rightarrow AB_a(XG(D_b(C)))$
8. $M \models_m Delegate(a, c, C(d, a, b, p, S)) \Rightarrow AB_b((XG(I_c(C)))U(satisfied(C) \vee breached(C) \vee canceled(C)))$
9. $M \models_m Delegate(a, c, C(d, a, b, p, S)) \Rightarrow AB_bF(satisfied(C))$

Theorem 1. *The debtor and creditor will never end up believing a commitment is still active after it has been discharged.*

Proof. When a commitment C is discharged, from definition 5.3.2, *Agent a* (debtor) believes that C will be inactive globally from the next moment onwards. From definition 5.3.6, *Agent b* (creditor) believes that C is inactive from the next moment onwards.

Similar proofs can be given when a commitment is canceled or released.

7 Example Uses of the BDI Commitment Formalism

We present examples of how our formalization can be used to explain, interpret, and model real-world multiagent systems. We use the travel agent example presented by Xing and Singh [13], where a customer contacts her travel agent to book a trip to a city with many hotels and airports. The travel agent requests airline and hotel clerks to make appropriate reservations and send confirmations to the traveler. The customer, travel agent, airline agent, and the hotel agent are all autonomous entities (*persons or their representative agents*).

When a customer contacts the travel agent to book a trip, the travel agent creates a commitment. Per *definition 5.1.2*, the travel agent `believes` that such a commitment will eventually be satisfied. Similarly, the customer `believes` that the travel agent's commitment will be satisfied eventually, which is consistent with *definition 5.1.7*. Also, per *definitions 5.1.3* and *5.1.8*, the travel agent `intends` to satisfy its commitment and the customer `desires` for that commitment to be satisfied. When the reservations are made and the customer is satisfied, the commitment is discharged. In accord with *definition 5.3.5*, the travel agent now `believes` it has satisfied its commitment, which becomes inactive. The customer, per *definition 5.3.7*, no longer `desires` for the trip to be booked again (unless he initiates a new instance of a trip-booking commitment).

Carrying this example further, consider a scenario where a customer assigns its commitment to another agent. Per *definitions 5.5.1* and *5.5.2*, we can see that the customer does not `desire` the travel agent to book a trip for him. Instead, the customer `believes` that the agent to whom the commitment was assigned `desires` that trip, which is explained by *definition 5.5.3*.

Consider the example from [6] of a travel agent who wishes to book an airline ticket to a certain destination, a rental car to use while there, and a hotel room in which to stay. The four scenarios discussed in [6] are (1) the travel agent wanting the passenger to fly on a particular day while still reserving the right to choose any flight on that day, (2) the car rental company offering a one-week free rental at a later time, (3) a hotel offering an electronic discount coupon that expires today, but text on the coupon states that it can only be used during a future spring break, and (4) the car rental company offering a warranty that cannot be used during the period in which the warranty is valid. The first two scenarios can be implemented directly with our formalism, as any conditions in such scenarios can be specified as the condition p in our commitment structure.

When there is a violation of time constraints similar to scenarios three and four, temporal operators in our condition p can capture the time constraints and show that the commitment cannot be satisfied. Our CTL* framework takes care of all the time constraints in a commitment and the BDI architecture captures the commitments in terms of the states of mind of the participating agents.

When an agent has more than one commitment, the only link or relationship between them has to be through the agent's mind. These commitments need to be consistent with the agents's internal beliefs, which our formalization helps to achieve. BDI can also be used to determine which of the several commitments an agent does first. For example,

Alice commits to pay *Bob* \$5
Bob commits to pay *Joe* \$5

Because *Bob* believes it will get \$5 from *Alice*, it then can form an intention to honor its commitment to *Joe*. The BDI system allows the agents to contend with multiple simultaneous commitments in the real world. As an example, if the beliefs and desires are not included in the model of the travel agent transaction scenario, the seller agent simply makes a commitment to the buyer agent based upon the ticket availability information it has received from the airlines.

Since desire is not modeled, the buyer and seller cannot barter, as they will make their commitments based purely upon the availability of tickets and money. If the seller has pricing from a single airline, it will make the commitment to sell, and the buyer will make the commitment to buy no matter what the price is as long as he has the money. The seller is not going to call other airlines, and the buyer is not going to call other travel agents due to lack of desires. Thus a better deal for both is not possible here.

If the desire of an agent is modeled, then the buyer's desire is to obtain the cheapest price and this will make it change its commitment to the seller. The commitment here is no longer "I will pay you whatever price you want," but it could be "I will buy the ticket only if it is the cheapest price."

On the other hand, the Seller's desire is to get the best price for the ticket, and the seller's commitment will be "I will you sell you the ticket only if you agree to pay cash within 24 hours and for credit I will charge you 10% extra." The seller has modified his commitment, because he has recognized the implicit threat in the buyer's commitment. The buyer has the desire to get the cheapest ticket as demonstrated by its commitment. If the seller does not modify its commitment, the buyer could go elsewhere. At the same time, to safeguard against a defaulted payment from the buyer and still make a profit, the seller is willing to offer or match the price if cash is paid.

Applying our formalization to the example, *Alice* commits to pay *Bob* \$5 means that *Alice* creates a commitment

Create(Alice, C(1,Alice,Bob,pay($5),S))
such that

1. *Alice* and *Bob* believe that commitment C will be active until it is either satisfied or breached or canceled or suspended.

$$AB_{Alice}((XG(active(C)))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)));$$
$$AB_{Bob}((XG(active(C)))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)))$$

2. For all paths, *Alice* and *Bob* believe that commitment C will eventually be satisfied.

$AB_{Alice}F(satisfied(C)); AB_{Bob}F(satisfied(C))$

3. For all paths from the next moment onwards, *Alice* intends the commitment C until it is either satisfied or breached or canceled or suspended.

$AXG((I_{Alice}(C))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)))$

4. For all paths, *Alice* believes that from the next moment onwards *Bob* desires commitment C until it is either satisfied or canceled .

$AB_{Alice}((XG(D_{Bob}(C)))U(satisfied(C)\vee canceled(C)))$

5. For all paths, *Bob* believes that from the next moment onwards *Alice* intends commitment C until it is either satisfied or breached or canceled or suspended.

$$AB_{Bob}((XG(I_{Alice}(C)))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)))$$

6. For all paths from the next moment onwards, *Bob* desires commitment C until it becomes inactive.

$AXG((D_{Bob}(C))U(\neg active(C)))$

When *Bob* commits to pay *Joe* $5, *Bob* creates a commitment

Create (Bob, C(2, Bob, Joe, pay($5), S))
such that

1. *Bob* and *Joe* believe that commitment C will be active until it is either satisfied or breached or canceled or suspended.

$$AB_{Bob}((XG(active(C)))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)));$$
$$AB_{Joe}((XG(active(C)))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)))$$

2. For all paths, *Bob* and *Joe* believe that commitment C will eventually be satisfied.

$AB_{Bob}F(satisfied(C));\ AB_{Joe}F(satisfied(C))$

3. For all paths from the next moment onwards, *Bob* intends the commitment C until it is either satisfied or breached or canceled or suspended.

$AXG((I_{Bob}(C))U(satisfied(C)\vee breached(C)\vee canceled(C)\vee suspended(C)))$

4. For all paths, *Bob* believes that from the next moment onwards *Joe* desires commitment C until it is either satisfied or canceled.

$AB_{Bob}((XG(D_{Joe}(C)))U(satisfied(C) \vee canceled(C)))$

5. For all paths, *Joe* believes that from the next moment onwards *Bob* intends commitment C until it is either satisfied or breached or canceled or suspended.

$AB_{Joe}((XG(I_{Bob}(C)))U(satisfied(C) \vee breached(C) \vee canceled(C) \vee suspended(C)))$

6. For all paths from the next moment onwards, *Joe* desires commitment C until it becomes inactive.

$AXG((D_{Joe}(C))U(\neg active(C)))$

Bob believes that commitment 1 will be satisfied eventually and he will get the \$5. He intends to get the money from *Alice* and pay that to *Joe* and thus satisfy commitment 2 (to pay *Joe* \$5). Using this formalization, the system can have rules, dependent on its requirements and available resources, to represent these intentions. As a simple example, *Bob* can have a rule such as

$ReceiveMoney(Alice, \$5) \Rightarrow PayMoney(Joe, \$5)$ (When money is received from *Alice*, pay that to *Joe*).

The above examples demonstrate how our formalization can be utilized to understand, explain, interpret, and model a real-world, commitment-centric, multiagent system. Our formalization is an improvement over the temporal logic approaches in [6,13], since it bridges BDI architectures and commitments.

8 Conclusion and Future Directions

Many real world systems are becoming cooperative. In a cooperative multiagent system, commitments represent agent associations and interactions, and a participant agent's beliefs, desires, and intentions about the commitments in which it is involved are critical to modeling agent behavior. With this formalization of commitments in terms of an agent's beliefs, desires, and intentions, we have provided the basic framework on which a more comprehensive commitment-driven decision theory can be developed. The advantage of this theoretical framework is that it blends two very robust and widely accepted theoretical frameworks that together can be utilized to model a cooperative multiagent system. These two frameworks are BDI_{CTL*} and commitments.

Our future research involves exploration of continuous commitments, how agents decide what to commit (earlier works on "capability" [8] can be integrated with commitments), when to cancel a commitment, how does a commitment "age," degree of commitment, and how can historical information of an agent's commitment adherence be utilized to predict its behavior. Commitment adherence and Sphere of Commitment can also be tied to *trust*. Moreover, with the help of either utility models or probabilities, a more comprehensive *commitment-driven decision theory* can be developed to model a cooperative multiagent environment expressively.

References

1. Castelfranchi, C.: Commitments: From Individual Intentions to Groups and Organisations. In: Proceedings of the Int. Conf. on Multi-Agent Systems 1996 (1996)
2. Cohen, P.R., Levesque, H.J.: Intention Is Choice with Commitment. Artificial Intelligence 42(2-3), 213–261 (1990)
3. Emerson, E.A.: Handbook of Theoretical Computer Science: Formal Models and Semantics, pp. 995–1072. MIT Press, Cambridge (1991)
4. Fasli, M.: On Commitments, roles and obligations. In: Central and Eastern European Conference on Multi-Agent Systems, pp. 93–102 (2001)
5. Jennings, N.R.: Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems. The Knowledge Engineering Review 8(3), 223–250 (1993)
6. Mallya, A.U., Huhns, M.N.: Commitments Among Agents. IEEE Internet Computing 7(4), 90–93 (2003)
7. Mallya, A.U., Singh, M.P.: An Algebra for Commitment Protocols. Autonomous Agents and Multi-Agent Systems 14(2), 143–163 (2007)
8. Padgham, L., Lambrix, P.: Agent Capabilities: Extending BDI Theory, pp. 68–73. AAAI/IAAI (2000)
9. Rao, A.S., Georgeff, M.P.: Modeling Rational Agents within a BDI-Architecture. In: Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning, pp. 473–484 (1991)
10. Singh, M.P.: Commitments Among Autonomous Agents in Information-Rich Environments. Modelling Autonomous Agents in a Multi-Agent World, 141–155 (1997)
11. Singh, M.P.: An Ontology for Commitments in Multiagent Systems: Toward a Unification of Normative Concepts. Artificial Intelligence and Law 7(1), 97–113 (1999)
12. Singh, M.P., Huhns, M.N.: Service-Oriented Computing: Semantics, Processes, Agents, pp. 363–370. Wiley, London (2005)
13. Xing, J., Singh, M.P.: Engineering Commitment-Based Multiagent Systems: A Temporal Logic Approach. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 891–898 (2003)

Towards an Open Negotiation Architecture for Heterogeneous Agents

Koen V. Hindriks, Catholijn Jonker, and Dmytro Tykhonov

EEMCS, Delft University of Technology, Delft, The Netherlands
{k.v.hindriks,c.m.jonker,d.tykhonov}@tudelft.nl

Abstract. This paper presents the design of an open architecture for heterogeneous negotiating agents. Both the system level architecture as well as the architecture for negotiating agents are provided. The main contribution of this paper is that it derives a precisely specified interface from these architectures that facilitates an easy integration of heterogeneous agents into the overall negotiation framework. The interface is defined as a set of adapters that allows for various levels of integration of agents into the system architecture. The functionality provided by the system architecture depends on the number of adapters that are implemented and used to connect an agent to this architecture, ranging from functionality to conduct a bilateral negotiation to functionality for computing agent internal performance measures such as the quality of an opponent model. The architecture is used as the basis of a competitive testbed which allows us to study various negotiating agents. The design yields a flexible negotiation framework that facilitates negotiating different domains potentially using different protocols whereas no details of the internal negotiating agent structure are enforced. An application of the framework is illustrated by integrating two agents from the literature.

1 Introduction

The boost of literature on negotiating agents and strategies of recent years is in line with the continuous advance of ecommerce applications, such as eBay, and Marketplace in which negotiations play a role. While the literature focuses on the development of ever more clever negotiation agents [6, 7, 11, 14, 17, 19], the actual use of these agents in ecommerce applications is prohibited by two factors: the inflexibility of the agents and the lack of ecommerce applications that are open to such agents [13].

By the inflexibility of the agents we refer to the fact that they are incapable of negotiating with arbitrary agents and incapable of negotiating on arbitrary subjects. The code created for agents introducing new strategies in the literature typically has been developed with respect to one or a few specific domains, and to run against other agents implemented by the same team [7, 14, 17, 19]. This is understandable, since the code is to provide evidence of the excellence of the strategy. As a consequence, however, these agents cannot participate in a generic negotiation environment where heterogeneous agents can interact with each other. Interaction between such agents is not feasible due to several problems such as the absence of a shared negotiation ontology, and the lack of support for generic interaction protocols. Open negotiation environments and testbeds reported so far, such as the Trading Agent Competition [2], propose ontologies for a specific domain or scenario. The shared negotiation ontology must be generic to be able to model arbitrary negotiation domains.

Current as well as newly developed negotiating agents are (will be) written by different teams that should be free to select the technology of their choice to build such agents. In practice it is not possible to impose a particular coding and design standard for developing negotiating agents. The applicability of such agents, however, depends on their ability to interact in order to negotiate. Both the inflexibility of the current state-of-the-art negotiating agents and the closedness of existing ecommerce applications warrants the specification of a well-defined and precisely specified interface that allows such agents to conduct a negotiation.

Previous work on resolving these issues has focused mainly on the specification of generic interaction protocols [4, 23]. Our aim has been to design and implement a negotiation framework that allows existing heterogeneous agents to negotiate and to analyze the results of such negotiation. The framework should be able to function as a *testbed* as well as provide the *enabling technology for integrating heterogeneous agents*. To this end, an approach must be developed that enables the integration of arbitrary agents and algorithms for automated negotiation into a generic negotiation system architecture. In particular, an open system architecture for heterogeneous negotiating agents is needed, as well as a conceptually simple and generic agent architecture, in order to clarify the *requirements* on an interface to connect arbitrary negotiating agents to an overall system architecture that supports (bilateral) negotiation. Our choice to introduce an overall system architecture thus is motivated by several considerations: (i) it can be used to create a principled design of an interface enabling heterogeneous negotiating agents to engage in negotiation, (ii) it may be used as a testbed as well as for defining particular standards used to define a negotiation problem, and (iii) it precludes the need to specify ad hoc agent-to-agent interfaces. The architecture and interface developed in this paper provides the basis for an implementation of a testbed for negotiating agents that includes a set of negotiation problems for benchmarking agents, a library of negotiation strategies, and analytical tools to evaluate an agent's performance and their strategies.

The paper is organized as follows. In Section 2 we propose an open architecture for heterogeneous negotiating agents and present a generic conceptual design of a negotiating agent architecture. Using this design an interface between agent and system architecture is specified. In Section 3 the adapters that are part of the interface are explained. The approach is illustrated by integrating two negotiating agents introduced in [17] and [19]. In Section 4 experiments are presented that demonstrate the usefulness of the environment as a testbed. Related work is discussed in Section 5. Section 6 concludes the paper.

2 Negotiation System and Agent Architecture

We introduce an architecture as a first step to a solution to the integration problem. The solution is applicable for integration of the existing agents as well as for the new agents that have not been implemented yet. This architecture has been implemented and provides the basis of our software negotiation framework. (This negotiation framework, user manuals, and a number of implemented negotiating agents can be downloaded from `http://mmi.tudelft.nl/negotiation`.)

Figure 1 illustrates the proposed architecture. The architecture is based on the analysis of the tasks of a generic negotiation environment. It represents a minimal but sufficient framework to enable integration of negotiation agents. The architecture consists of *four main layers* introduced below, a *human bidding interface*, and a *negotiating agent architecture.* An *interaction layer* is required to define and define the negotiation protocol and enable communication between agents. *An ontology layer* is needed to provide the actual functionality needed to define, specify and store a negotiation domain, the preferences of the negotiating agents. The architecture can be used for education and training of humans in negotiations. For that purpose, *a graphical user interface layer* provides options to create a negotiation ontology, defines agent preferences, allows human user(s) to participate in a negotiation, and review performance and benchmark results of agents that conducted a negotiation. *An analytical toolbox* is required to use the system as a research tool and organize tournaments. It provides a variety of tools to analyze the performance of agents and possibly internal quality measures related to e.g. the quality of an opponent model.

The overall architecture is introduced here to identify the main integration points where adapters are needed to connect a negotiating agent to this architecture. For the purpose of this paper, the human bidding interface is not relevant. The agent architecture itself identifies common components of a negotiating agent but is not intended to provide a comprehensive analysis of such architectures or go beyond the current state of the art [3, 8, 14]. This architecture may be instantiated with various software agents, which we illustrate below.

2.1 Negotiation System Architecture

Graphical User Interface. The graphical user interface enables a user to define the negotiation game, i.e. the parameters of the negotiation, the subject or domain of negotiation, and preferences of agents (which also means that the preferences a human should take into account can be predefined). This interface does not introduce any integration points that should be part of the interface to integrate negotiating agents into the negotiation environment.

Negotiation Domain. A negotiation domain is a specification of the objectives and issues to be resolved by means of negotiation. It specifies the structure and content of bids or offers exchanged, and of any final outcome or agreement (see also Fig. 4 and 5 below). An outcome determines a specific value for each issue, or, alternatively, only for a subset of the issues. Objectives allow to define a

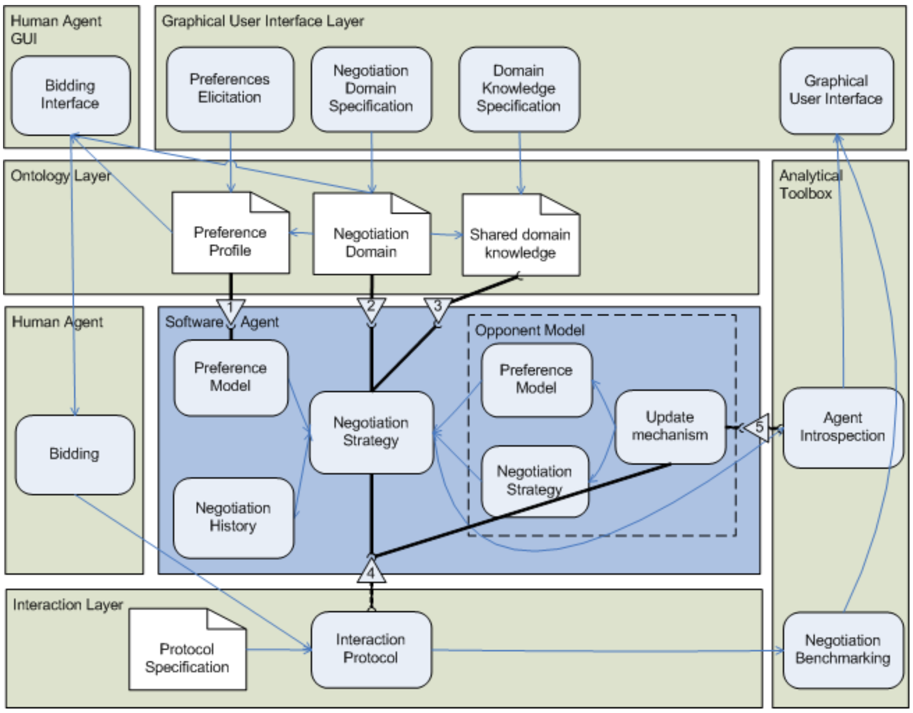


Fig. 1. The Open Negotiation System Architecture

tree-like structure with either other objectives again or issues as children, in line with [22]. Various types of issues are allowed, including discrete enumerated value sets, integer-valued sets, real-valued sets, as well as a special type of issue called *price* issue. Additionally, a specification of a negotiation domain may introduce constraints on acceptable outcomes. For example, costs associated with a particular outcome may not exceed the available budget of the agent.

Preference Profile. A preference profile specifies the preferences regarding possible outcomes of an agent. It can be thought of as a function mapping outcomes of a negotiation domain onto the level of satisfaction an agent associates with that outcome. The structure of a preference profile for obvious reasons resembles that of a domain specification (see also Fig. 4 and 5 below). The tree-like structure allows to specify relative priorities of parts of the tree. This allows, for example, to ensure that all issues relating to travelling combined are weighted equally as all issues relating to the actual stay at a particular location.

Shared Domain Knowledge. In a *closed* negotiation an agent is not informed about the preferences of its negotiating partner. In that case an agent can at best use a reconstruction (using e.g. machine learning techniques) of these preferences to decide on the negotiation move it should do next. It is typical, however, that with a domain comes certain public knowledge that is shared and can be used to obtain a better negotiation outcome. For example, common preferences such as preferring early delivery over later (though not always the case) may be common knowledge in a given domain. Such knowledge allows agents to compute the preferences of their negotiation partner e.g. using the time interval between two dates. This type of knowledge, labelled *shared domain knowledge*, is modelled explicitly as a separate component that can be accessed by all negotiating agents.

Interaction Protocol. The interaction layer manages the *rules of encounter* or *protocol* that regulate the agent interaction in a negotiation [18]. Any agent that wants to participate in such a negotiation protocol must accept and agree to conform to these rules. An interaction protocol specifies which negotiation moves and what information exchange between agents is allowed during a negotiation. Interaction protocols are implemented in the negotiation environment as a separate component to allow the use of a variety of protocols [4]. The current version of the negotiation environment supports the alternating offer protocol [18], that allows a generic communication between the agents. The protocol is illustrated in Figure 2. A protocol can also dictate the exchange of complete package deal proposals or allow instead the exchange of partial bids. The layer also manages deadlines, or timeouts that may be set by the environment.

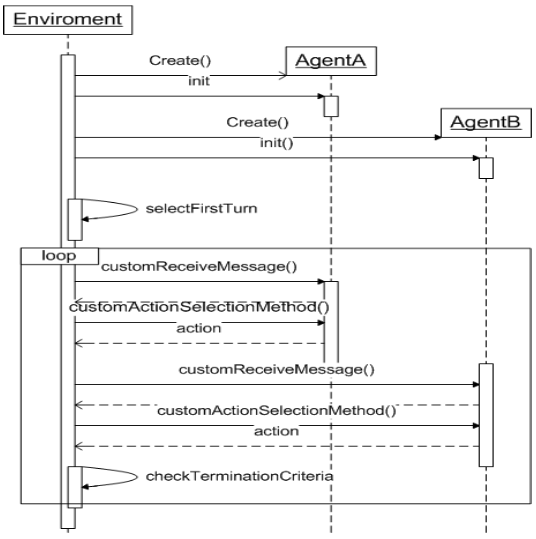


Fig. 2. A sequence diagram of the interaction protocol

The alternating offer protocol is not the only protocol used in the negotiation research. Therefore, the interaction protocols are implemented in the negotiation environment in a separate component to allow the use of a variety of protocols [4]. Implementation of a new interaction protocol in the negotiation environment is relatively easy task and has no or minimal effect on the agent code.

Analytical Toolbox. The analytical toolbox layer of the architecture contains a set of statistical analysis methods to perform an outcome analysis on negotiation sessions as introduced and discussed in e.g., [10, 22]. Furthermore, the toolbox contains methods for the analysis of dynamic properties of negotiation sessions as discussed in e.g., [10]. The methods for both outcome and dynamics analysis were used to produce a number of performance benchmarks for negotiation behaviour and for the agent components [11]. The analytical toolbox uses the optimal solutions [21], such as the Pareto efficient frontier, Nash product and Kalai-Smorodinsky solution for the negotiation outcome benchmarking. The benchmarks in the negotiation system can be used to analyze the performance of opponent modelling techniques, the efficiency of negotiation strategies, and the negotiation behaviour of the agent. The result of the analysis can help researchers to improve their agents. The output of the analytical toolbox is presented by means of visualization (e.g., see 6).

2.2 Software Agent

The software agent component highlighted by the use of a different colour in Figure 1 is a generic component that can be instantiated by heterogeneous software agents. The components specified as part of a software agent in Figure 1 are part of the *conceptual design* of such agents but do not need to be actually present or identifiable as such in any particular software agent. These components do not introduce any design requirements for negotiating agents (although they could be used as such, see also [3]). Instead these components are introduced to identify *integration points* of agents with the system architecture. Five of such integration points, also referred to as *adapters*, have been identified.

The *preference model* component models the agent's preferences with respect to the negotiation outcomes. For example, the agents introduced in [11, 17, 19] use *utility functions* to represent preferences. Preferences however can be modeled by other structures, such as *ordinal rankings*. The *negotiation strategy* is the core component of a negotiating agent. This component makes decisions about the acceptance of an opponent's offer, ending a negotiation, and sending a counter-offer, using various tactics to generate such counter-offers [6]. The *negotiation history* component maintains the negotiation history, i.e. bids exchanged between agents, and can be used by the negotiation strategy component. It can also have a history records about earlier negotiations, the outcomes, identities of the opponents, and even opponent models. In repetitive negotiations with the same opponents this information can be used to improve negotiation performance of an agent by adapting the negotiation strategy and improving the opponent model.

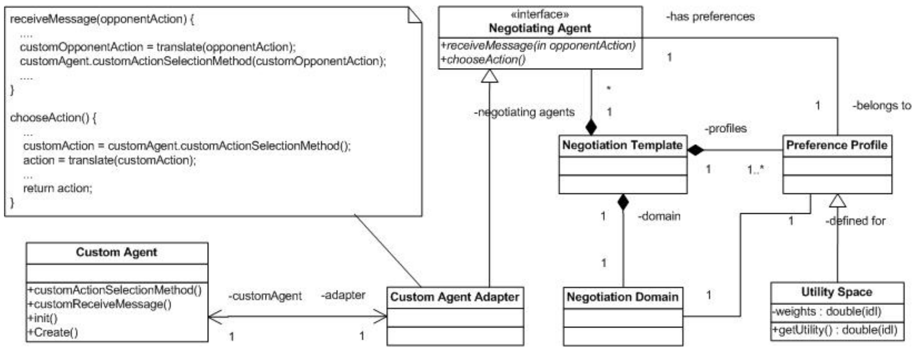
In a typical negotiation setup preferences of the negotiating parties are private [22]. However, the efficiency of a negotiation strategy can be significantly improved by using information about opponent preferences [24]. Thus, an important component of a negotiating agent is an *opponent model*. Our generic component consists of three subcomponents: a preference model, a negotiation strategy, and an update mechanism. The component *preference model* contains representations of the preferences of the current and previous negotiating opponents. Typically, since the opponent's preferences are assumed to be private, the information stored in the component has a degree of uncertainty. The component *update mechanism* is used to interpret offers received from an opponent and to update the probability distribution associated with the preferences of an opponent. The purpose of the component *negotiation strategy* in the opponent model is to predict negotiation moves of the opponent. This knowledge can be used in the negotiation strategy to improve the efficiency of an agent's own offers and increase the chance of acceptance of an offer by the opponent. Models of the opponent's preferences and strategy are typically learned by the agent from negotiated agreements and offers exchanged in a negotiation [11, 14, 24].

3 Interface and Adapters

To integrate heterogeneous negotiating agents in a single negotiation framework, their implementation has to be aligned with respect to the identified integration points of Figure 1. Alignment by redesign of an agent typically requires significant programming efforts and may cause back-compatibility problems. The number of adapters between agent to be developed in an *ad hoc* enviroment is quadratic in the number of agents: every pair of heterogeneous agents requires two adapters, one at each side. Wrapping agents and connecting them to a common framework requires only one adapter per agent, a number that is linear in the number of agents to be integrated. To minimize programming effort we propose a set of *adapters* or wrappers which need to be implemented once for each agent, and use software design patterns to develop these adapters [16]. From the five integration points identified, 3 must be implemented to be able to negotiate with another agent, including a negotiation domain adapter, a preference profile adapter and an interaction protocol adapter. Implementing the shared domain knowledge and agent introspection adapter provide additional functionality useful for more realistic negotiations, as well as for benchmarking agent performance.

In order to evaluate the effectiveness of our framework for integrating heterogeneous negotiating agents, we have integrated two existing agents from the literature, the *QO Agent* from [17] and the agent based on fuzzy modelling techniques [19] labelled *FBM* here. The integration of the agent should have no or minimal consequences for the performance of the agent. In order to validate that the integration did not affect the performance the integrated agent was evaluated and compared with the original implementation using the negotiation problems provided with that implementation. The results obtained did not show that the performance of the agents was significantly affected. Below we present the details and guidelines for implementing the adapters. Due to space limitations we cannot provide all details but only provide some specific findings regarding the integration of the *QO agent*.

Interaction Protocol Adapter. The negotiation framework provides a skeleton Java class, called *Negotiating Agent*, to facilitate the implementation of a custom-made negotiating agent (see Figure 3). This class implements basic



```
receiveMessage(opponentAction) {
  ....
  customOpponentAction = translate(opponentAction);
  customAgent.customActionSelectionMethod(customOpponentAction);
  ....
}

chooseAction() {
  ...
  customAction = customAgent.customActionSelectionMethod();
  action = translate(customAction);
  ...
  return action;
}
```

Fig. 3. A UML specification of the interaction protocol adapter

functionality of the agent such as the agent initialization, and the loading of a negotiation domain and preference profile, etc.

One of the most important tasks of this class is to ensure that a custom-made agent will comply with the negotiation protocol. The NegotiatingAgent class declares several methods that must be implemented in an agent. These methods are called by the system architecture during a negotiation to inform an agent about its opponent's last action and to allow the agent to respond.

To integrate an existing agent in the negotiation framework we have used the Object Adapter design pattern. Figure 3 shows the adopted design pattern for the negotiating agent. In line with the pattern definition a Custom Agent Adapter class is added that is inherited from the Negotiating Agent class The receiveMessage() and chooseAction() methods of the adapter use the translation routines of the negotiation domain adapter.

Key to the successful integration of an existing agent is understanding the original code to a sufficient degree to understand the main information flows and interaction patterns. The main problem is the significant amount of time that is needed to analyze agent code to gain this insight. In particular, it is important to identify the agent's methods (a) that evaluate and interpret opponent bids and (b) that decide on the agent's next action. Moreover, differences in a protocol used by one agent from that of another require choices to be made as to what protocol to use in the negotiation framework. As an example, the protocol used by the original *QO Agent* is different from the alternating offers protocol and we chose to use the alternating offers protocol in our experiments.

Negotiation Domain and Shared Domain Knowledge Adapters. This adapter must be able to interpret the information about the domain such as the number of issues, type of the issues and the values of the issues. Figure 4 shows a class diagram of the negotiation domain implementation of our negotiation framework. The Negotiation Domain class is a composition of a set of issues represented by the super class Issue. All classes for the specific issue types inherit from the Issue super class. Issues can be grouped into a hierarchical structure using the Objective class. As we explained earlier, our system provides four

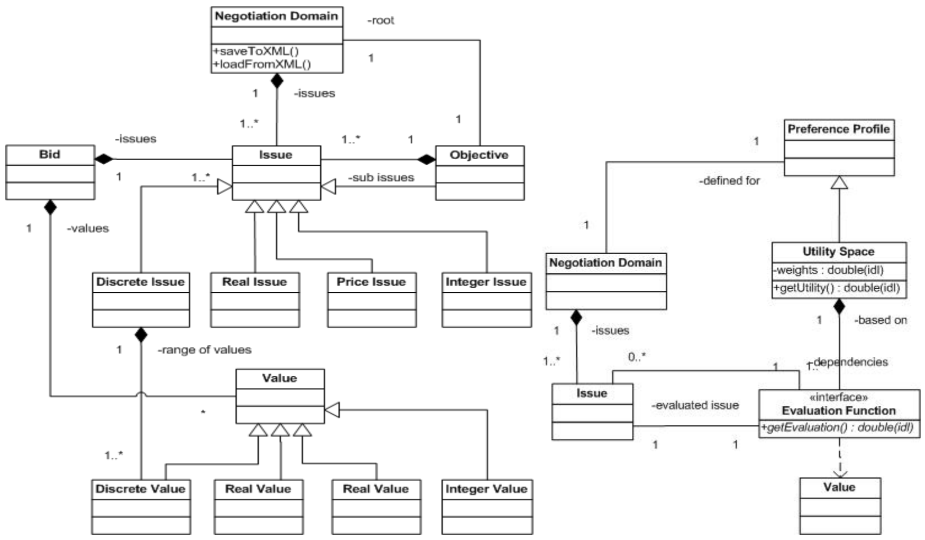


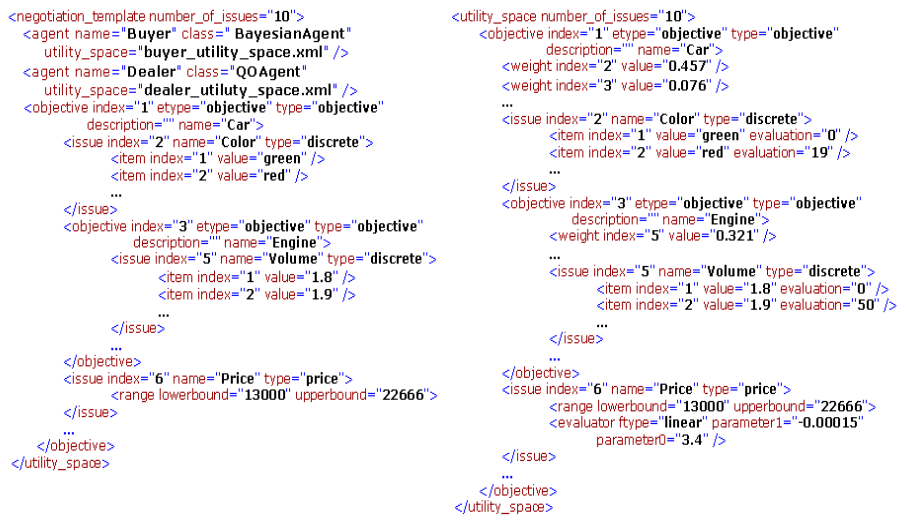
Fig. 4. Class diagram of the negotiation domain (left) and preference profile (right)

different types of issues: discrete, real, integer, and price issue (see the left part of the Figure 4 for the corresponding classes). Issues and corresponding values are bounded in the Bid class. An object of the Bid class represents one of the possible outcomes of a negotiation domain. The system implements a number of consistency checks to ensure that a bid is valid given the domain specification.

The adapter is implemented by two routines that translate the domain model provided by the negotiation system into the internal representation of the agent and vice versa. These routines are used to load a negotiation domain into the agent, interpret an incoming proposal from an opponent and generate negotiation moves.

A negotiation domain is represented by using a particular negotiation ontology. The negotiation ontology we use is specified in terms of XML files [1], which is widely-accepted file format for information exchange. Specific tags in the XML file correspond to each of the various Java classes used to store a negotiation domain. For example, the Objective class is represented with the tag labelled "objective" in the XML file that contains a negotiation domain specification (see the left part of the Figure 5). This tag can nest child tags such as other objectves and issues to represent the hierarchy as explained above. The *type* attribute of the *issue* tag specifies the type of the issue, such as discrete, real, integer. Discrete issues are defined by the *item* tags. An *item* tag defines a possible value of the issue in the *value* attribute. Intervals for the real and integer issues are defined using the lowerbound and upperbound of the *range* tag. The shared domain knowledge is also encoded in the negotiation domain XML file. The semantics of the domain knowledge must be interpreted by the agent itself, but is in fact supplied by means of the Negotiation Domain class which defines an XML document object model (DOM) [1].

Obviously, the routines to be developed mapping the ontology of the negotiation framework onto that of the negotiating agent and vice versa need to take the expressivity of the resepective ontologies into account. Only those parts of the ontologies can be mapped on each other that express the same meaning. Therefore the implementation of the negotiation domain adapter requires a careful analysis of the negotiation ontology of the original agent implementation first. Using this analysis the functions that translate negotiation concepts from the ontology used by the agent and the ontology used by the system architecture (see Fig. 5) have to be implemented. This procedure rather straightforwardly could be applied to the *QO agent*, which uses issues that can take discrete values, and uses plain text files to store a negotiation domain. Since discrete issues are one type of issue allowed by the negotiation system it is easy to define the required adapter methods.The method the *QO agent* needs to read a negotiation domain from an XML file (the format used by the system architecture) was wrapped up in a negotiation domain adapter.



```
<negotiation_template number_of_issues="10">
  <agent name="Buyer" class=" BayesianAgent"
     utility_space="buyer_utility_space.xml" />
  <agent name="Dealer" class="QOAgent"
     utility_space="dealer_utiluty_space.xml" />
  <objective index="1" etype="objective" type="objective"
          description="" name="Car">
       <issue index="2" name="Color" type="discrete">
             <item index="1" value="green" />
             <item index="2" value="red" />
             ...
       </issue>
       <objective index="3" etype="objective" type="objective"
                description="" name="Engine">
             <issue index="5" name="Volume" type="discrete">
                   <item index="1" value="1.8" />
                   <item index="2" value="1.9" />
                   ...
             </issue>
             ...
       </objective>
       <issue index="6" name="Price" type="price">
             <range lowerbound="13000" upperbound="22666">
       </issue>
       ...
    </objective>
</utility_space>
```

```
<utility_space number_of_issues="10">
    <objective index="1" etype="objective" type="objective"
             description="" name="Car">
       <weight index="2" value="0.457" />
       <weight index="3" value="0.076" />
       ...
       <issue index="2" name="Color" type="discrete">
             <item index="1" value="green" evaluation="0" />
             <item index="2" value="red" evaluation="19" />
             ...
       </issue>
       <objective index="3" etype="objective" type="objective"
                description="" name="Engine">
             <weight index="5" value="0.321" />
             ...
             <issue index="5" name="Volume" type="discrete">
                   <item index="1" value="1.8" evaluation="0" />
                   <item index="2" value="1.9" evaluation="50" />
                   ...
             </issue>
             ...
       </objective>
       <issue index="6" name="Price" type="price">
             <range lowerbound="13000" upperbound="22666">
             <evaluator ftype="linear" parameter1="-0.00015"
                   parameter0="3.4" />
       </issue>
       ...
    </objective>
</utility_space>
```

Fig. 5. XML specification of a negotiation domain (left) and preference profile (right)

Preference Profile Adapter. This adapter must be able to interpret the preferences of the agent as specified in Figure 4. The current implementation of the negotiation system operates with a preference structure based on utility functions. Other preferences modeling techniques, such as an ordinal ranking of the outcomes can be implemented by inheriting from the Preference Profile class. The utility space class in the negotiation system calculates the utility of an outcome as a weighted sum of the evaluations values of the individual issues, i.e. it implements linearly additive utility functions. The same type of utility functions are used by the *QO Agent*.

The preference profile, as the negotiation domain, can be saved as an XML file (see Figure 5). The structure of a preference profile XML file is similar to and extends the negotiation domain XML file with information about issue evaluators and their associated weights (priorities). The type of the evaluator is specified using the *type* attribute of the objective and issue tags. For example, for a discrete issue utility values are specified in the *evaluation* attribute of the *item* tag that represents the value of that issue. Consistency of a preference profile given a corresponding domain is checked automatically when it is loaded from the XML file by the negotiation system.

The procedure for implementing the preference model adapter is similar to that of implementing the negotiation domain adapter. As before, the representation of the agent's preferences in the original implementation need to be analyzed. In addition, one should verify whether the structure of the utility space and evaluation functions of the negotiation framework can be used to model the structure of the preferences in the original implementation. Since these aspects match for the *QO agent* the adapter could be implemented without much problems.

Agent Introspection Adapter. The negotiation system architecture can be used as a testbed and research tool because it provides a number of benchmarks and tools to analyze negotiation performance [11]. To facilitate such analysis, an introspector is provided by the negotiation system. Negotiating agents can notify this introspector about a variety of events, such as an update of the opponent model, the selection of a next negotiation move, etc. The introspector must be allowed access to some of the internal structures of an agent such as its preference profile, its opponent model, and its negotiation history to be able to fully perform its function. This access is required to compute e.g. metric distances between an opponent preference model constructed by the agent and the actual preferences of the opponent.

We use the Observer pattern which is an event-driven design pattern to implement the introspector functionality. An agent needs to register with the introspector that plays the role of Observer. Components of the agent then need to notify the introspector when a corresponding event appears, which are subsequently logged and analyzed to obtain benchmarks.

Dedicated code must be written to be able to have the introspector compute some relevant performance measuresfor a particular agent. For example, for an agent that tries to learn the opponent's preferences measures related to the quality of learning as proposed in [12] can be computed. In order to do so, it is most important to locate those places in the agent code that can be used effectively for notifying the introspector to (re-)calculate the performance measures.

Lessons Learned. The expressive power of the ontologies available for the specification of the negotiation domain and preference profiles were sufficient to express all possible options to define a domain and profile for both the *QO agent* as well as the *FBM agent*. The preference profiles of both agents are implemented as utility functions and the evaluation functions used by the agents to evaluate the values of the issues were also already present in our system architecture. The implementation of the adapter for the interaction layer, however, was more complicated than expected. The main reason is that the interaction protocol used by the original *QO agent* extends the alternating offers protocol since it allows additional types of messages to be exchanged between agents, such as threats. Moreover the localization of the core functions of the agent needed by the interaction protocol adapter determined most of the integration efforts. Ideally, therefore, an existing agent is integrated in close cooperation with its original developer. This is not an issue, however, if agent are implemented from scratch.

4 Experiments

One of the purposes of the proposed architecture is to allow for integration of heterogeneous agent and to facilitate comparison of their negotiation effectively as a testbed, and can be used to perform experiments with various negotiation domains, preference profiles and negotiating agents. The framework thus contributes to automated negotiating agents research by providing a tool that is able to provide new insights about such agents.

A tournament is a typical experimental setup for negotiating agents [10] that allows to measure success of an agent compared to the performance of the other. In addition, it is a useful tool to study the influence of various factors on the negotiation performance [12]. The analytical toolbox of the framework can generate a tournament setup given a set of agents, negotiation domains and preference profiles.

A small and simple negotiation problem, called "Party" [12], is used to analyze the performance of the *QO agent* within our negotiation framework. This domain has been created for negotiation experiments with humans, which also explains its rather limited size, including only 5 discrete issues with 5 possible values each (totaling to 3,125 possible outcomes). All of the issues are unpredictable [12], i.e. there is no shared domain knowledge. The preference profiles for the experiment were selected randomly from a set of 30 profiles created by human participants in a previously performed experiment. Since the *QO agent* needs 3 profiles as possible models of the opponent's preferences to be able to learn that profile, it was provided with the real profile of its opponent and two additional profiles that were randomly selected from all profiles.

In the experimental setup the *QO agent* negotiated against the Bayesian agent introduced in [11]. The Bayesian agent uses a learning algorithm using a Bayesian learning technique to build a model of the opponent's preferences. The techniques learns the necessary probabilities over a set of hypotheses about the evaluations and weights of the issues. Structural assumptions about the evaluation functions and weights are made to decrease the number of parameters to be learned and to simplify the learning task. Only one experiment was run for each combination of agents due to deterministic nature of the negotiation strategies of both agents.

Figure 6 presents the results of the negotiation experiment. The charts show the space of all possible negotiation outcomes. The axis represent the utilities

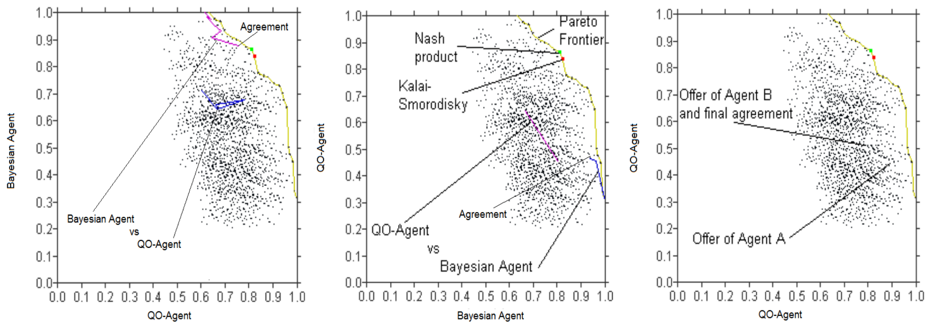


Fig. 6. Negotiation dynamics for the Party domain

of the outcomes with respect to the utility functions of the negotiating agents. The charts show the negotiation paths of the agents marked by arrows with the names of the agents.

The Bayesian agent starts with an offer that has maximum utility. It tries to learn the opponent preferences from the offers it receives and uses this model when it makes a concession towards the opponent. As a result, it stays close to the Pareto Efficient frontier. The *QO agent* in this domain has more difficulty to propose efficient offers. *This is a result of limitation of the opponent model of the agent.* The *QO agent* accepts an offer of the Bayesian agent as soon as such an offer has a utility level for the *QO agent* that is higher then utility of the *QO agent*'s own offer.

The other agent integrated into our negotiation system is the *FBM* agent introduced in [19]. The *FBM* agent was tested in a setup where it has to negotiate against the Bayesian agent about a single issue defined on real values ranging from 10 to 30. The original *FBM* agent is designed for negotiations where agents can exchange fuzzy proposals. The implementation of the *FBM* agent we used is able to negotiate about one-issue negotiations but can be extended for multi-issue negotiations. The agent adopts time dependent negotiation tactics from [6] and, thus, always makes concessions towards opponent. The offers are defined using two values: the peak value and the stretch of the offer. The preference profiles of the agents used were in complete opposition: the *FBM* agent wants to minize the value of the issues and the *Bayesian* agent tries of maximize it. In the experiments we performed, the β parameter that defines whether an agent makes bigger concessions in the beginning of the negotiation (Conceder) or at the end (Boulware) was varied, see Table 1.

In a single issue negotiation all negotiation outcomes are Pareto effient. The most important aspect of the negotiation strategy in a single issue negotiation is how fast one conceeds to the opponent. As a result, for $\beta > 1$ the *FBM* agent implements a Conceder tactic and the *FBM* agent undeperforms with respect to the *Bayesian* agent that makes linear concessions in this case because no moves towards the Pareto frontier are possible. When the *FBM* agent employs a Boulware tactic ($\beta < 1$) the *Bayesian* agent starts conceeding significantly and the result is a much lower utility for the *Bayesian* agent.

Table 1. Utility values of the FBM and Bayesian agents

Agents	Utility						
	β=0.02	β=0.1	β=0.5	β=1	β=2	β=10	β=50
FBM Agent	0.898	0.897	0.734	0.585	0.449	0.193	0.060
Bayesian Agent	0.102	0.103	0.266	0.415	0.551	0.807	0.940

5 Related Work

There is a large body of related work available. Due to lack of space we cannot provide a complete overview but discuss specific approaches and examples of negotiation frameworks that allow us to clearly position our own work.

Generic frameworks for negotiation. A range of quite different negotiation frameworks exist in the literature, including frameworks for (i) automated negotiating agents as well as (ii) negotiation support systems that provide electronic support for human negotiations. Within the first class a distinction can be made between argumentation-based systems, e.g. [8, 20], and heuristic-utility based systems, e.g. [10, 14, 17, 19]. The framework introduced and implemented belongs to the heuristic-utility based class of systems, though in principle it should be possible to use the framework for argumentation-based negotiation as well. Negotiation Support Systems (NSS) refer to systems that assist the process of human communication in negotiation, see, for example, [5, 15]. For example, in the Althena project (`www.althenasoft.org`) users can build content models, but the system does not support by means of predefined structures, repositories of content models, interaction support, or the selection of bidding strategies. Similarly, our framework, through a graphical user interface, allows users to create preference profiles, but significant extensions are needed in order to provide similar negotiation support useful for humans.

Architectures for negotiating agents. The main focus in the literature on negotiating agent architectures, e.g. [3, 8, 14], is on the descriptive, structural and behavioural specification but not on the design of and requirements associated with interfaces. The system and agent architecture presented here are used specifically to obtain these interface requirements.

Negotiation ontologies. Our work is related to work on negotiation ontologies to the extent that we need to define a language that can be used by heteregeneous agents to exchange offers. The language that we have used, based XML schemas, has been expressive enough to be able to integrate the *QO agent* and *FBM agent*. Our efforts to set up a negotiation ontology for the architecture proposed thus are motivated primarily by experience gained in practice. Related work about ontologies such as [4, 23] focus more on protocol than domain ontologies, and can be viewed as complementary. In future work our framework will be extended to handle negotiation ontologies for protocols as well.

Testbeds and Trading Competitions. There is a variety of testbeds and trading competitions, but most are based on auction models instead of bilateral negotiation. In contrast, the framework introduced here provides a testbed that can be used to evaluate the behavior and performance of automated negotiating agents in bilateral negotiation. Moreover, most of the available testbeds are based on a specific domain, whereas we believe that there is a need for multi-issue bargaining testbeds which facilitate negotiation about various domains. A system somewhat similar to ours is the Neg-o-Net system [9]. Neg-o-Net is a generic agent-based computational simulation model for conducting multi-agent negotiations concerning resource and environmental management decisions, and includes a number of negotiation algorithms as well as agent models. However, the system is developed specifically to investigate environmental management.

6 Conclusion and Future Work

In this paper we have defined a clear interface for integrating bilateral negotiating agents into a (competitive) testbed. We have shown our approach is a viable and realistic one by demonstrating the actual integration of two negotiating software agents according to the "recipe" discussed in the paper. Some initial experimental results were provided to illustrate that by using the proposed environment we were able to easily obtain some new results and gain new insights on the current state of the art in the area of automated negotiation.

The negotiation environment described and developed based on the principles and specifications introduced here implements an open system architecture for such agents. The interface and adapters to connect agents to the negotiation environment have been clearly specified which enable an easy integration of heterogeneous negotiating agents. The actual use of this environment in, for example, ecommerce applications based on bilateral negotiation, however, is still significantly beyond its current main function as a testbed environment. Future research should pave the way to such applications, which would involve among others providing agents with the capability to propose a domain of negotiation, and to define the rules of the negotiation game (i.e., protocol selection). Additional research on ontologies for negotiation are required to make this feasible; for example, we cannot currently forumulate associated constraints on the domain of negotiation that must be satisfied for an agreement to be acceptable. More technically, components for web integration as well as extensions of adapters need to be developed, e.g., in order to handle more generic ontologies.

Acknowledgements

We would like to thank Raz Lin and Zaynab Raeesy for providing us with the code of their negotiating agents.

References

1. Extensible markup language (xml), http://www.w3.org/XML
2. The trading agent competition, http://www.sics.se/tac

3. Ashri, R., Rahwan, I., Luck, M.: Architectures for negotiating agents. In: The 3rd Int./Central And Eastern European Conf. on Multi-Agent Systems (2003)
4. Bartolini, C., Preist, C., Jennings, N.: A generic software framework for automated negotiation. Technical report, HP Labs (2002)
5. Bellucci, E., Zeleznikow, J.: A comparative study of negotiation support systems. In: Proceedings of HICSS (1998)
6. Faratin, P., Sierra, C., Jennings, N.R.: Negotiation decision functions for autonomous agents. Int. Journal of Robotics and Autonomous Systems 24(3-4), 159–182 (1998)
7. Faratin, P., Sierra, C., Jennings, N.R.: Using similarity criteria to make negotiation trade-offs. Journal of Artificial Intelligence 142(2), 205–237 (2003)
8. Geipel, M.M., Weiss, G.: A generic framework for argumentation-based negotiation. In: Klusch, M., Hindriks, K.V., Papazoglou, M.P., Sterling, L. (eds.) CIA 2007. LNCS (LNAI), vol. 4676, pp. 209–223. Springer, Heidelberg (2007)
9. Hales, D.: Neg-o-net - a negotiation simulation test-bed. Technical Report CPM-03-109, CPM, April, Published as part of the FIRMA workpackage 3 report (2002)
10. Hindriks, K., Jonker, C.M., Tykhonov, D.: Negotiation dynamics: Analysis, concession tactics, and outcomes. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, pp. 427–433 (2007)
11. Hindriks, K., Tykhonov, D.: Opponent modelling in automated multi-issue negotiation using bayesian learning. In: Proceedings of the AAMAS 2008 (2008)
12. Hindriks, K., Tykhonov, D.: Towards a quality assessment method for learning preference profiles in negotiation. In: Proceedings of the AMEC 2008 (2008)
13. Jennings, N.R., Sycara, K., Wooldridge, M.: A roadmap of agent research and development. Journal of Autonomous Agents and Multi-Agent Systems (1998)
14. Jonker, C.M., Robu, V., Treur, J.: An agent architecture for multi-attribute negotiation using incomplete preference information. Journal of Autonomous Agents and Multi-Agent Systems 15(2), 221–252 (2007)
15. Kersten, G.E., Lai, H.: Negotiation Support and E-negotiation Systems: An Overview. Group Decision and Negotiation 16(6), 553–586 (2007)
16. Larman, C.: Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edn. Prentice-Hall, Englewood Cliffs (2004)
17. Lin, R., Kraus, S., Wilkenfeld, J., Barry, J.: Negotiating with bounded rational agents in environments with incomplete information using an automated agent. Artificial Intelligence Journal 172(6-7), 823–851 (2008)
18. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press, Cambridge (1994)
19. Raeesy, Z., Brzostwoski, J., Kowalczyk, R.: Towards a fuzzy-based model for human-like multi-agent negotiation. In: Proc. of the IEEE/WIC/ACM Int. Conf. on Intelligent Agent Technology, pp. 515–519 (2007)
20. Rahwan, I., Ramchurn, S.D., Jennings, N.R., McBurney, P., Parsons, S., Sonenberg, L.: Argumentation-based negotiation. The Knowledge Engineering Review 18(4), 343–375 (2004)
21. Raiffa, H.: The Art and Science of Negotiation. Harvard University Press (1982)
22. Raiffa, H., Richardson, J., Metcalfe, D.: Negotiation Analysis: The Science and Art of Collaborative Decision Making. Harvard University Press (2003)
23. Tamma, V., Phelps, S., Dickinson, I., Wooldridge, M.: Ontologies for supporting negotiation in e-commerce. Engineering Applications of Artificial Intelligence 18(2), 223–236 (2005)
24. Zeng, D., Sycara, K.: Bayesian learning in negotiation. International Journal of Human Computer Systems 48, 125–141 (1998)

Incrementally Refined Acquaintance Model for Consortia Composition

Jan Doubek, Jiří Vokřínek, Michal Pěchouček, and Martin Rehák

Agent Technology Group, Gerstner Laboratory
Department of Cybernetics, Czech Technical University in Prague
duby.jan@seznam.cz, {vokrinek,pechouc}@labe.felk.cvut.cz

Abstract. This paper presents a specific contracting algorithm that contributes to the process of distributed planning and resource allocation in competitive, semi-trusted environments. The presented contraction algorithm is based on incrementally refined acquaintance models (IRAM) of the actor that provide the right set of approximate knowledge needed for appropriate task decomposition and delegation. This paper reports on empirical evaluation of the IRAM algorithm deployment in consortia formation domain.

1 Introduction

This work focuses on a technique for distributed consortium formation with limited knowledge sharing. The consortia formation is based on a negotiation between independent self-interested providers. The providers can offer several services and the goal is to find the best suitable composition of providers to cover required set of services.

The targeted domain organizes multi-party interaction in the environments that are:

– **non-centralized** and with flat organizational structure [R1] – the existence of a central coordination is minimal and the information about the skills of actors, resource availability, knowledge and goals is distributed,
– **multi-party involvement** [R2] – the final project cannot be implemented in isolation by a single actor, consortium composition can be initiated by several actors simultaneously,
– provides **partial knowledge sharing** [R3] – the actors in the environment are motivated to keep a substantial part of their private planning knowledge and resource availability information undisclosed.

Due to the presented requirements, the consortia cannot be evaluated centrally and the service allocation to the individual providers has to be negotiated. The goal is to minimize the interactions with the providers to the necessary minimum to reduce their private knowledge disclosure and simultaneously ensure the quality of the solution. To fulfill those demands we have designed presented algorithm and acquaintance model representation and provide experimental evaluation.

This algorithm contributes to the process of distributed planning and resource allocation in competitive, semi-trusted environments. The presented algorithm is based on incrementally refined acquaintance models (IRAM) – the model that the actor is maintaining about potential collaborators [1].

2 Problem Statement

The consortium composition can be represented as distributed state-space search through all the potential consortia in the environment with limited information sharing. Let us denote R as a requester agent, A_t as a set of services that R requests to fulfill the task t. Furthermore we have agent P_j, as a provider agent offering a certain set of services A_j^{max}, where $j \in \{1 \dots n\}$ and

$$F_j(A_j) : \{A_j \subset A_j^{max}\} \tag{1}$$

is pricing function for agent P_j to provide set of services A_j. When asked agent P_j sends back just the price value.

The problem is then to acquire the optimal price

$$c(A_t) = \min \sum_{j=1}^{n} F_j(A_j) \rightleftharpoons \{A_j \subset A_j^{max}; \bigcup_{j \in \{1 \dots n\}} A_j = A_t\} \tag{2}$$

In this paper we then focus to find optimal vector of sets $(A_1, \dots, A_n)$ as a decomposition of task t, where the overall price $c(A_t)$ is minimal. Set of all vectors that satisfies task the condition $\bigcup_{j \in \{1 \dots n\}} A_j = A_t$ will be referenced as a *Deal Space* - DS_t.

In our model we have made several assumptions:

– **Fixed price** – the price of a particular subset of services is fixed during algorithm run.
– **Tasks are independent** – a provider is capable of delivering same services during all negotiation even if he was contracted for some services in the previous tasks.
– **Non-increasing partial price** – a provider constructs a $F_j(A_j)$ as aggregated price from prices for individual services that are hidden to the requester. We assume the individual services price to be non-increasing in reference to increasing $|A_j|$.

3 IRAM-Based Consortium Formation

We have designed a straightforward decomposition mechanism that finds the optimal decomposition given the right objective function and a complete information about provider's resource availabilities. The decomposition algorithm is polynomial and easy to construct (see [2]). Its behavior, however, worsens strongly with lower quality of information about the provider's prices stored in the requestors' acquaintance models (containing a subset of the deal space). The most efficient approach in fully cooperative communities would be if the requestor queries all the providers and reconstructs the deal space for all services provided by all actors prior to computing the optimal a contract.

As this is not possible in the environment compliant with the requirements R1 and R3, the requestor needs to approximate such knowledge with only partially available information. We are proposing *incrementally refined acquaintance model* (IRAM) algorithm for handling partial knowledge sharing and private knowledge disclosure [1].

This approach has been evaluated and compared with the another method of provider prices estimation - the well-known Chebyschev Polynomials approximation method.

3.1 Acquaintance Model

The acquaintance model can have a number of forms [3], [4]. In this particular application the acquaintance model is understood as function that predicts actor responses to a particular *call-for-proposals* (`CFP`) type of message. We represent the *acquaintance model* (am) as a mapping from a set of $\mathcal{P}(A_j^{max})$ possible subsets asked from the provider P_j to a 1 dimensional real-value space representing cost $\mathcal{C}$.

$$\mathcal{F}_j^{am} : \mathcal{P}(A_j^{max}) \rightarrow \mathcal{C} \tag{3}$$

Let us discuss several properties of an acquaintance model. The *fixed point* is such a mapping among the actor, single service and a particular cost that is based on exact information acquired from the communication with the specific actor. In a fixed point as_x^σ

$$\mathcal{F}_j^{am}(as_x^\sigma) = f_j(s_x, |A_j|, pc_j) \text{ where } |A_j| = \sigma \tag{4}$$

where $f_j(s_x, |A_j|, pc_j)$ represents the price contribution of presence of service s_x in A_j to the total price of the entire set $F_j(A_j)$.

Provided that the fixed points of the acquaintance model are collected in a set $\Delta(\mathcal{F}_j^{am})$, we define the *size of the acquaintance model* $\delta(\mathcal{F}_j^{am})$ the amount of the fixed points in the acquaintance model as follows:

$$\delta(\mathcal{F}_j^{am}) = |\Delta(\mathcal{F}_j^{am})|. \tag{5}$$

Various approximation functions have been used in the acquaintance models, e.g. [2]. In our model we have selected the pairwise constant approximation. The **unknown** price of service s_y in subset with size $|A_j|$, equals to the closest bigger known fixed point in means of the size of the containing subset$|A'_j| : s_y \in A'_j$

$$\mathcal{F}_j(s_y, |A_j|, pc_j) = \mathcal{F}_j(s_y, |A'_j|, pc_j) \text{ if } |A_j| \lessdot |A'_j|, \tag{6}$$

provided that the symbol $\lessdot$ represent the smallest bigger value.

The *error of the acquaintance model* - $\epsilon(\mathcal{F}_j^{am})$ - represents how well does the acquaintance model capture real capability of the providers. Error of the acquaintance model is a dual quantity to the *quality of the acquaintance model*. There can be a number of ways how the error can be related to the quality. We only require that with a monotonic increase of quality the error decreases and vice versa.

We represent the error of the acquaintance model as a sum of the differences between the real costs and the information on costs provided by the acquaintance model.

$$\epsilon(\mathcal{F}_j^{am}) = \sum_{p,j} |\mathcal{F}_p^{am}(A_{p,j}) - f_p(A_{p,j}, pc_j)| \tag{7}$$

the index p goes through all possible subsets of A_j, and j goes through all partners.

As said before, the reason why we use the acquaintance models for contracting is that we are motivated by minimizing the unwanted knowledge disclosure during interaction (requirement R3). Each interaction represents disclosure of private information. By `CFP` the actors disclose their inability to perform a task as well as their intention to do so. By a response to `CFP` the agents disclose information about availability of particular resources. It is evident that with rising $\delta(\mathcal{F}_j^{am})$, the acquaintance model is more exact and thus provides better information (i.e. lower $\epsilon(\mathcal{F}_j^{am})$). Better acquaintance model managed to reduce communication (and thus private knowledge disclosure) during the negotiation between the actors. However, bigger $\delta(\mathcal{F}_j^{am})$ (and thus smaller $\epsilon(\mathcal{F}_j^{am})$) required substantial interaction during the acquaintance model construction phase where lots of unwanted information may have been disclosed.

The IRAM algorithm is balancing the size and the quality of the acquaintance models. In order to evaluate performance of the IRAM algorithm we have developed a reference algorithm that is working with a similar acquaintance model, constructed prior negotiation. Both algorithms are based on distributed state-space search using negotiation between actors. As a negotiation protocol, we use the *competitive contract-net protocol* [5], but any protocol that enables iterative contract negotiation can be used.

3.2 IRAM Algorithm

The run of this algorithm for one particular task t is started with the initiation phase. All providers are contacted for every single service and for maximal subset of services from task t. The IRAM model am is constructed using the closest bigger known fix-point approximation (see eq. 6). The model is represented by sets of prices for specific service and pricing function settings (see eq. 1). The price is set blank when is not known, and thus is calculated from other fixed points.

The algorithm constructs the Deal Space and evaluates it with the prices from $\mathcal{F}_j^{am}$. The cheapest consortium $Cons^{best}$ is selected and the providers are requested for appropriate services. Offered prices are then integrated into the IRAM model. The deal space is then reevaluated and the cheapest consortium is selected. If the new consortium is composed from fixed points (represents

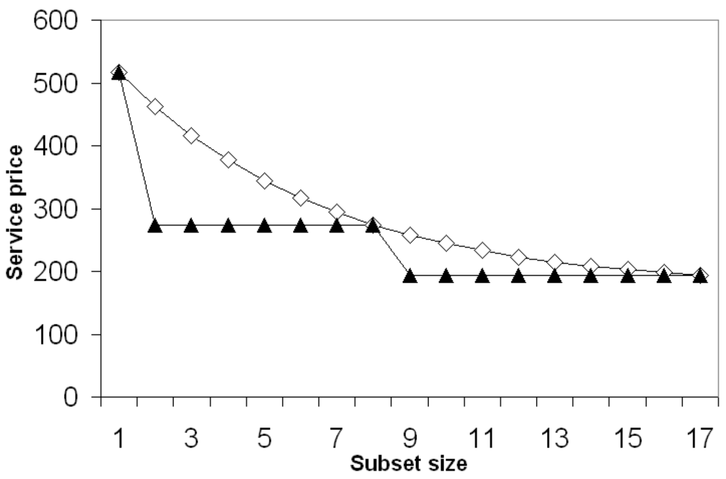


Fig. 1. IRAM model(labeled with triangles) for one service according to s_x, fixed points in 1,8,17. Approximated function is labeled by diamonds.

real price of the consortium – no price approximation), this we understand as optimum. Request for this consortium will lead to the exact same information and due to eq. 6 the algorithm has converge to the optimum. The phases of IRAM can be seen below.

Initialization. It is necessary to know at least two fixed points of acquaintance model for specific service from each provider, for proper functionality of IRAM algorithm. So if the algorithm have not these from previous contracts, it obtains them in the first iteration. Due to this fact the amount of communication is considerably higher regarding to following iterations. Preferably we choose the single service data and maximum provider coverage data (A_j^{max}). Single service provides us data needed for proportional price reconstruction necessary due to aggregated price. And the max coverage data gives us the lowest possible prices from provider needed for approximation.

Iteration Phases. The iteration phases represent processes that follow each other in further negotiation stage.

– Contacting the best known consortium given by acquaintance model
– Updating IRAM model by the received responses
– Reevaluating the acquaintance model
– Sorting the deal space by total consortium price
– Termination condition evaluation

Termination Condition. The algorithm is iterating (contacting and updating model) till the best evaluated consortium consists of fixed points only (e.g. no part of consortium has estimated evaluation)

The steps of IRAM algorithm can be seen in Figure 2.

```
1 Construct deal space $DS_t$.
2 Send CFP($s_i$) for all $s_i \in t$ to all providers $P$.
3 Update $am$ according to received responses.
4 Send CFP($A_p^{max}$) to all providers $P$.
5 Update $am$ according to received set of responses $A_p$.
6 Select $Cons^{best}$ from $DS_t$ evaluated by $am$.
7 If $Cons^{best} \subset \bigcup_{j \in P}(\Delta \mathcal{F}_j^{am})$ then terminate algorithm.
8 Send CFP($A_{p,j}$), where $\bigcup_{j=1\ldots n} A_{p,j} = Cons^{best}$ to providers $1\ldots n$.
9 Goto 5.
```

Fig. 2. IRAM algorithm steps

3.3 Properties of IRAM

The presented IRAM algorithm is sound and complete. The proof of completeness of the algorithm is made through conversion of whole idea to A^* algorithm [6], where the nodes of searched space are individual consortia from DS, and edges represent inclusion (or exclusion) of one provider to a consortium. This representation corresponds to *Coalition Structure graph* [7].

The heuristics of A^* is then based on acquaintance model approximation, all of the nodes are priced particularly by real prices (fixed-points) and by computed prices given by acquaintance model. The price of the consortium $Cons$ is represented by

$$c(Cons) = g(Cons) + h(Cons),$$

$$\text{where } g(Cons) = \sum_{A_j \subset \Delta(\mathcal{F}_j^{am})} (\mathcal{F}_j^{am}(A_j)) = \sum_{A_j \subset \Delta(\mathcal{F}_j^{am})} F_j(A_j),$$

$$\text{and } h(Cons) = \sum_{A_j \subset \mathcal{P}(A_j^{max})/\mathcal{P}(\Delta(\mathcal{F}_j^{am}))} \mathcal{F}^{am}(A_j). \tag{8}$$

The $g(Cons)$ represents price of the of subsets from $Cons$ that is known from previous negotiations (the fixed-points) and $h(Cons)$ is the unknown price of the subsets from $Cons$ estimated by acquaintance model.

The non-increasing individual pricing function assumption causes

$$s_x \in A_{j,1}; s_y \in A_{j,2}; s_y = s_x; |A_{j,1}| \leq |A_{j,2}|$$

$$\Rightarrow f_j(s_x, |A_{j,1}|, pc_j) \geq f_j(s_y, |A_{j,2}|, pc_j) \tag{9}$$

According to eq. 6 and 9 the $h(Cons)$ is always equal or lower then the real price of this subset, so $h(Cons) \leq h^*(Cons)$ and the eq. 8 is admissible heuristics of A^* algorithm. Since the IRAM is based on exploration of the best candidates evaluated by eq. 8 the algorithm provides the features of A^* algorithm [6].

3.4 Reference Algorithm

The presented approach has been empirically validated by comparison with the state-of-the-art algorithm with behavior similar to IRAM algorithm. Generally the IRAM method is used to approximate unknown pricing functions of partners and determine the direction of future negotiation. For the benchmarks purposes we have implemented a reference algorithm based on deployment of known Chebyshev polynomials [8], previously used for modeling of sellers production pricing function.

The *Chebyshev polynomials* exactly Chebyshev polynomials of the first kind are defined as

$$T_n(x) = \cos n \arccos x \tag{10}$$

Due to its orthogonality with respect to the weight $w(x) = (1-x^2)^{-1/2}$ in the interval [-1,1] are widely used for approximation, in most cases they are more effective than Taylor's. The Chebyshev polynomial state can be represented by the recursion formula: $T_{n+1} = 2xT_n(x) - T_{n-1}(x)$, where $T_0 = 1; T_1 = x$. Further information can be obtained in [9].

The approximation itself is made through computing weights (c_k). They represent the contribution of every Chebyshev polynomial to the resulting function. The complete approximation formula is

$$f(x) \approx \sum_{k=0}^{N-1} c_k T_k(x) - \frac{1}{2}c_0 \tag{11}$$

The defining weights are reconstructed from approximated known points $(x_k, f(x_k)), k = 1..M$ with formula

$$c_j = \frac{2}{M} \sum_{k=1}^{M} f(x_k) T_j(x_k) \tag{12}$$

The values of x_k should be mapped to $[-1, 1]$.

3.5 Implementation

As mentioned above, searched approximation method was selected according to initial environmental conditions similar or equal to IRAM's. In case of Chebyshev polynomials the conditions matched exactly. The implementation of the IRAM algorithm was just slightly modified in the field of approximation and all the other interfaces (like communication, consortium construction and evaluation) were left untouched. For a proper explanation of its deployment, let us explain the adaptation of Chebyshev polynomials to the environment. The first two samples (fixed points), needed in general for every approximation, are gathered from the initiation phase of negotiation (same as IRAM). The deal space is then evaluated from Chebyshev approximation of the pricing functions. The final condition is also same as in IRAM. If the same solution is evaluated as the best in two following iterations it is returned as result.

As mentioned previously, the only alteration (from the IRAM algorithm) in the optimal consortium search was the approximation process. Just like in IRAM there is a pricing function model for every particular service from every provider, this model consists of set of weights (eq. 11), the number of weights depends on the number of Chebyshev polynomials used. The number is also correlated with approximation quality (will be further described). The origin of the polynomials is the same for all approximations thus is stored as a constant. The exploiting of the incoming information takes place during the weight computation, which is performed as a result of collecting each new price information.

4 Experiments

The key contribution of the presented paper is in empirical evaluation of the presented algorithm. We will be analyzing the behavior of IRAM in relation to the reference algorithm presented above.

For presenting the contribution of IRAM we construct a market model that contains a set of four providers P, one simple *requester* that requests providers for set of 17 tasks $S = t_1 \dots t_{17}$ using IRAM and reference algorithm for comparison.

In our experiments we randomly generated 4 providers, where everyone of them was capable of delivering 14 services from total 18 services, this setting was chosen due to computation requirements. The max size of *acquaintance model* is $\delta_{max}(\mathcal{F}_j^{am}) = 65532$ possible proposals i.e. *fixed points*, and average *deal space* size in one task is $\langle |DS_t| \rangle = 8777$. The individual pricing functions were randomly generated as follows. We generate uniform distribution set of prices $UP = (bp_1 \dots bp_s)$ in defined range $(200, 600)$ for base price bp of every service from U. Then we create one random value in range $d_j \in (0.6, 1)$ for particular provider p_j that represent the discount in price according to the number of total services asked $|A_j|$. From discount is then computed margin value m_j

$$m_j = 1 + 0.05 * ((1 - d_j)/0.8) * ((1 - d_j)/0.8 + 1)/2. \tag{13}$$

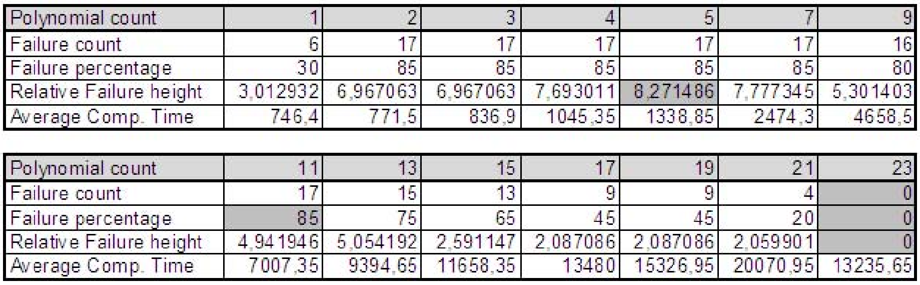
The price $f_{i,j}$ of single service s_i is then derived from base price bp_i and total service asked $|A_j|$ as follows.

$$f_j(s_i) = m_j bp_i d^{|A_j|-1} + 0.5 m_j bp_i \tag{14}$$

Then the pricing function corresponds to eq. 1 Every provider then responds only with this price when is asked for some services.

4.1 Quality of a Model

As shown in [8], the quality of a approximation rises with the number of polynomials used. It can be shown that for a approximation of a polynomial function of a certain degree d the number of Chebyshev polynomials needed for exact approximation is also d. However in our case the pricing functions (eq. 1) are little bit different to be specified exactly like a polynomial. Therefore we run quality tests to find the right Chebyshev polynomial count. The results are shown in figure 3.



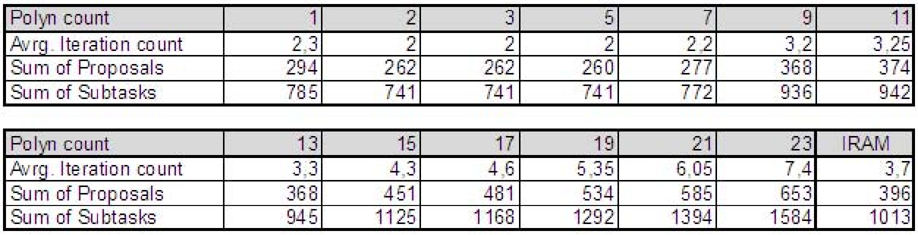
Polynomial count	1	2	3	4	5	7	9	11	13	15	17	19	21	23
Failure count	6	17	17	17	17	17	16	17	15	13	9	9	4	0
Failure percentage	30	85	85	85	85	85	80	85	75	65	45	45	20	0
Relative Failure height	3,012932	6,967063	6,967063	7,693011	8,271486	7,777345	5,301403	4,941946	5,054192	2,591147	2,087086	2,087086	2,059901	0
Average Comp. Time	746,4	771,5	836,9	1045,35	1338,85	2474,3	4658,5	7007,35	9394,65	11658,35	13480	15326,95	20070,95	13235,65

Fig. 3. Chebyshev approximation performance with different polynomial count

The scenario of the test is performed by 6 partners with 17 services each from 34 possible services. They are gradually asked for 20 tasks composed of 9 services. The pricing functions specifications are the same as usual.The size of the deal space is then according to the setting 19683. There is a set of tests on the same setting just with different count of Chebyshev polynomials used. Measured variables was *failure count* (finding the wrong optimum), *failure percentage*, *average relative failure* height (to the price of optimum) and *computational time* (on the same hardware setting in ms). The table shows us that the approximation has interesting numbers in one polynomial case, which represents the linear approximation with one straight line. The other results worsens with rising polynomial number until the peak at 11 polynomial for failure count and 5 for average relative failure height. The computational time is rising due to computing of higher and higher powers. By the polynomial count of 23 we can se ideal approximation of our pricing functions represented by 0 failures. So this setting we can use as a proper benchmark for IRAM in this particular environment.

4.2 Benchmarking IRAM vs. Chebyshev

Finally we can unveil the key comparison of Chebyshev method and IRAM. For the first set of tests we simply use the setting described in previous section in order to illustrate major differences. The measured variables are *average Iteration count*, *sum of asked proposals* and *sum of asked services*. The Figure 4 providing measured variables with IRAM's results in the last column, clearly shows that with the same level of quality (zero failures) the shared information (gathered and paid) from partners is 50 % bigger in negotiation than using Chebyshev approximation. To be equal in information requirements we have to accept 85 % chance to have 5 % failure with Chebyshev of the order 11. Just for clear comparison of the *average computational time* for one IRAM negotiation was 859,8 ms comparing to 13235,65 ms of Chebyshev of the order 23. The time results have just relative information value because the implementation of Chebyshev polynomial computation can be slightly upgraded, which is not the key object of this paper. For the next experiments we will use the Chebyshev polynomial of the order 11 and Chebyshev polynomial of the order 23 for benchmarking with IRAM's results.



Polyn count	1	2	3	5	7	9	11
Avrg. Iteration count	2,3	2	2	2	2,2	3,2	3,25
Sum of Proposals	294	262	262	260	277	368	374
Sum of Subtasks	785	741	741	741	772	936	942

Polyn count	13	15	17	19	21	23	IRAM
Avrg. Iteration count	3,3	4,3	4,6	5,35	6,05	7,4	3,7
Sum of Proposals	368	451	481	534	585	653	396
Sum of Subtasks	945	1125	1168	1292	1394	1584	1013

Fig. 4. IRAM vs. Chebyshev comparison

Another key comparison which can be shown on multiple task solving scenarios, is the progress of the harvested data utilization. Due to incrementally better and larger model sizes we can obtain the solution with lower negotiation requirements. Results of this experiments are shown in figures 5 and 6. In the first one there is the number of fixed points requested from the partners in every negotiation. This value has the meaning of shared information among the partners. We can see a notable computational overhead needed for Chebyshev of the order of 23 in the beginning of the negotiations. This trend then gets significantly better than IRAM numbers with increasing number of task solved. This is caused by bigger (thus better) Chebyshev of the order 23 model. The Chebyshev of the order 11 has almost the same data. It was chosen because of its equal data requirements as IRAM. On this particular data we can see progressive IRAM's character in the single task formation problem, i.e. the model is build from scratch. In the first step of the graph IRAM outperforms Chebyshev by 50 %. In unknown environments or in highly dynamic cases this capability become very useful. In the second figure we shown the sizes of each model (see eq. 5) and how this size continually grows with the number of negotiations.

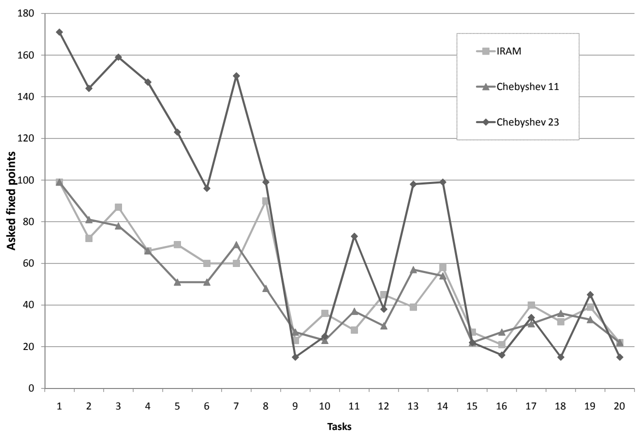


Fig. 5. IRAM vs. Chebyshev 11 and Chebyshev 23 asked fixed points in particular negotiations

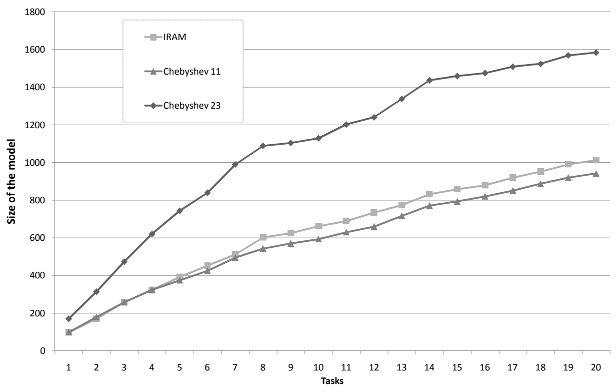


Fig. 6. IRAM vs. Chebyshev 11 and Chebyshev 23 in size of the models

Again, we can see the overhead needed by Chebyshev of the order 23. It comes with the robustness of 23 Chebyshev polynomials and its requirements for initial data to construct the proper approximation. Trends for all of the algorithms are almost the same. With an increasing number of negotiations they tend to converge to certain bound where the models have mapped all interesting areas of partner's pricing functions. In those regions IRAM shows significantly lower need for collected information to provide the best solution.

As we can see, the deployment of IRAM algorithm in this type of domains brings significant improvement in performance, not just in information sharing field but with its simple implementation even in computational requirements.

5 Conclusion

The paper also presents a specific algorithm for distributed task delegation and resource allocation in semi-trusted multi-actor communities - the Incrementally Refined Acquaintance Model (IRAM). This algorithm is based on incrementally maintained social knowledge of the service requestor about the service providers. The novelty of the presented approach is in the fact that social knowledge is used even if very imprecise and it is gradually refined by means of unsuccessful attempts to contract.

The presented IRAM algorithm allows consortia composition with respect to defined requirements, mainly non-centralized approach and minimization of private knowledge disclosure. In environments with a certain degree of dynamics, IRAM's decent information requirement in initiative phase of building the model brings it a significant benefit in comparison with classical approximation approaches (Chebyshev). It can be outperformed in longer sets of negotiations due to more voluminous models representing the opponents.

High dynamics of the environment causes devaluation of acquaintance model and it can lead to incorrect solution. In such dynamic environment, the IRAM algorithm has to start building the model from scratch for every new task. In one deal scenario, the presented IRAM algorithm still provides quick convergence to the optimum with low communication and thus low private knowledge disclosure. Another option is limit the validity of the information obtained and reconstruct part of the model only.

Acknowledgment

The work is funded by the European Commission's FP6 programme within the projects Collaborative Process Automation Support using Service Level Agreements and Intelligent Dynamic Agents in SME clusters (PANDA), IST-No.027169. The research is also partly funded by the Ministry of Education, Youth and Sports of the Czech Republic grant No. MSM 6840770038 and No. 1M0567.

Any opinions expressed in this paper are those of the author(s)/organization and do not necessarily reflect the views of the European Community. The Community is not liable for any use that may be made of the information contained herein.

References

1. Pěchouček, M., Mařík, V., Bárta, J.: Role of acquaintance models in agent's private and semi-knowledge disclosure. Knowledge-Based Systems 19, 259–271 (2006)
2. Pěchouček, M., Lerch, O., Bíba, J.: Iterative query-based approach to efficient task decomposition and resource allocation. In: Klusch, M., Rovatsos, M., Payne, T.R. (eds.) CIA 2006. LNCS (LNAI), vol. 4149, pp. 258–272. Springer, Heidelberg (2006)
3. Cao, W., Bian, C.-G., Hartvigsen, G.: Achieving efficient cooperation in a multi-agent system: The twin-base modeling. In: Kandzia, P., Klusch, M. (eds.) Cooperative Information Agents. LNCS (LNAI), vol. 1202, pp. 210–221. Springer, Heidelberg, Heidelberg (1997)
4. Mařík, V., Pěchouček, M., Štěpánková, O.: Social knowledge in multi-agent systems. In: Luck, M., Mařík, V., Štepankova, O. (eds.) Multi-Agent Systems and Applications. LNCS (LNAI). Springer, Heidelberg (2001)
5. Vokřínek, J., Hodík, J., Bíba, J., Vybíhal, J., Pěchouček, M.: Competitive contract net protocol. In: van Leeuwen, J., Italiano, G.F., van der Hoek, W., Meinel, C., Sack, H., Plášil, F. (eds.) SOFSEM 2007. LNCS, vol. 4362, pp. 656–668. Springer, Heidelberg (2007)
6. Dechter, R., Pearl, J.: Generalized best-first search strategies and the optimality af A*. J. ACM 32(3), 505–536 (1985)
7. Sandholm, T.: Distributed Rational Decision Making. In: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence, pp. 201–258. MIT Press, Cambridge (1999)
8. Saha, S., Biswas, A., Sen, S.: Modeling opponent decision in repeated one-shot negotiations. In: AAMAS 2005: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, pp. 397–403. ACM, New York (2005)
9. Rivlin, T.: Chebyshev Polynomials: From Approximation Theory to Algebra and Number Theory. Wiley-Interscience, Chichester (1974)

Towards a Monitoring Framework for Agent-Based Contract Systems

Noura Faci, Sanjay Modgil, Nir Oren, Felipe Meneguzzi,
Simon Miles, and Michael Luck

Department of Computer Science, King's College London,
Strand, London WC2R 2LS, UK
noura.faci@kcl.ac.uk

Abstract. The behaviours of autonomous agents may deviate from those deemed to be for the good of the societal systems of which they are a part. Norms have therefore been proposed as a means to regulate agent behaviours in open and dynamic systems, and may be encoded in electronic contracts in order to specify the obliged, permitted and prohibited behaviours of agents that are signatories to such contracts. Enactment and management of electronic contracts thus enables the use of regulatory mechanisms to ensure that agent behaviours comply with the encoded norms. To facilitate such mechanisms requires monitoring in order to detect and explain violation of norms. In this paper we propose a framework for monitoring that is to be implemented and integrated into a suite of contract enactment and management tools. The framework adopts a non-intrusive approach to monitoring, whereby the states of a contract with respect to its contained norms can be inferred on the basis of messages exchanged. Specifically, the framework deploys agents that observe messages sent between contract signatories, where these messages correspond to agent behaviours and therefore indicate whether norms are, or are in danger of, being violated.

1 Introduction

Interactions in systems composed of heterogeneous and self-interested agents are inherently unreliable, requiring some form of societal control to bind these interactions. The introduction of norms has been proposed to address this need in such systems [8,1], allowing for open societies of autonomous agents that are, nevertheless, regulated to some degree. Such norms are usually specified using deontic concepts, including the notions of obligations, permissions and prohibitions that govern or direct agent behaviour in multi-agent systems. By incorporating sets of these norms into a formal document representation, it is possible to define electronic contracts, which mirror the the paper versions exchanged between businesses today, and offer the possibility of dynamic, runtime enforcement of *contract party* agent behaviours to ensure compliance with norms.

This work is situated in the context of the CONTRACT project[1] that aims to develop frameworks, components and tools that make it possible to model, build, verify

[1] www.ist-contract.org

and monitor distributed electronic business systems on the basis of dynamically generated, cross-organisational contracts which underpin formal descriptions of the expected behaviours of individual agents and the system as a whole. In this paper we report on ongoing development of the project's framework for *monitoring* electronic contracts.

The fact that agents are to varying degrees autonomous means that their behaviours may deviate from those prescribed by norms; that is, agents may violate norms. Thus there are requirements for monitoring norm violations during run-time contract-based governance of agent behaviours [14]. Monitoring has also been extensively studied in a Web Services context (e.g., [6,12,13]). However, monitoring of contracts specified as Service-Level Agreements (SLA), focuses on quality of service metrics rather than on the behaviours of contractual entities. We adopt the latter perspective in this paper; one that allows for planning-oriented detection and analysis of norm violation by agents. Two types of monitoring have been adopted in multi-agent systems:

1. In *corrective monitoring* (e.g., [5]) violations are detected as they occur, and the results of analysis of these violations are used to instigate corrective measures. Such measures might include the imposing of punishments on those contract parties that violate norms, thus motivating future compliance (as well as possibly compensating contract parties injured by non-compliance). These punishments may be manifest in the form of less favourable terms offered to the violating parties when re-negotiating contracts. Furthermore, the very fact that agents are being monitored, and thus the *threat* of punishment, may itself motivate compliance.
2. In *predictive monitoring* (e.g. [18]) norm violations are predicted and actions are specified to avoid violation. The obvious advantage is that the business process can continue uninterrupted by remedial measures.

Our focus in this paper is on corrective monitoring based on run-time observation of agent behaviours. However, we also discuss how the monitoring of agent behaviours can be used for predictive purposes.

Approaches to monitoring of agent behaviours can also be distinguished according to whether they adopt an *intrusive* or *overhearing* approach. In the intrusive approach (e.g., [2,4,16,9]), the mental states of agents are assumed to be available for inspection. Agents communicate their states to monitors that subsequently interpret the behaviours of agents. Intrusive approaches thus make the design of agent-based systems more complex, and rely heavily on the compliance of agents to communicate the required data. We therefore adopt the *overhearing* approach [5] in which messages exchanged among agents are observed, and behaviours are inferred from these messages.

The paper is organised as follows. Section 2 briefly describes the CONTRACT project's representation of electronic contracts, their contained norms, and the architecture for enactment and managing of contracts. Sections 3, 4 and 5 then describe the primary contribution of this paper: we propose a multi-agent framework for non-intrusive monitoring of electronic contracts, whereby the *states* of a contract with respect to its contained norms can be inferred on the basis of messages exchanged. In this way one can recognise whether a norm is currently in force, in danger of being violated, and whether a norm is in fact violated or complied with. We believe that our framework is the first to adopt such an approach to monitoring of electronic contracts.

Section 3 focuses on the entities and associated information flows that comprise the monitoring aspects of the architecture, while Section 4 then characterises the behaviour of the architecture's agents in terms of the messages these agents receive and send. Section 5 then describes the mapping from the norms expressed in a contract to a representation suitable for monitoring, and how violations are detected based on the processing of these representations and the observed exchange of messages between contract party agents. Section 6 looks forward to future work; in particular, work on generation of explanations for norm violations, and representation of danger states that indicate an increased likelihood of norm violation and thus enable predictive monitoring. Finally, Section 7 concludes and discusses related work.

2 Norms and Architecture

The CONTRACT project encodes contracts as XML documents consisting of normative clauses that are essentially declarative specifications of agent behaviours. Associated with these documents are ontologies that describe and define background concepts and terms. These contract documents and associated ontologies are more fully described in [15], but the XML encoding of normative clauses is the primary input to the monitoring architecture. Here, we briefly review our underlying model of norms that ground specification of a contract's normative clauses.

Norms can be classified into *obligations* (what should be done), *prohibitions* (what should not be done) and *permissions* (what is allowed to be done). Norms affect *target* agents that agree to abide by the norms contained in a contract, which apply under certain circumstances. Note that these circumstances may occur multiple times during a contract's lifetime. For example, an obligation to keep a fire door shut (the obligation's goal state or *condition*) takes force whenever the door is opened. Whenever such triggering (*activating*) circumstances arise, an instantiated version of the norm parameterised by those circumstances begins to take effect on the target agent's behaviour.

More formally, norms are tuples of the form:

$$(NormType, NormActivation, NormCondition, NormExpiration, NormTarget)$$

where $NormType \in \{\text{obligation}, \text{permission}, \text{prohibition}\}$, $NormActivation$ denotes the conditions under which the norm is activated (triggered), and $NormCondition$ denotes the goal or state that:

- must be brought about by the $NormTarget$ in the case of an obligation;
- may be brought about by the $NormTarget$ in the case of a permission; or
- must not be brought about by the $NormTarget$ in the case of a prohibition.

Finally $NormExpiration$ denotes the conditions under which the norm is no longer in force.

Example 1. Consider the following obligation $\mathbf{Ob}_{Del}$ on an agent AgX to deliver $GoodsZ$ to agent AgY within one week of receiving the order from AgY:

- $NormType$ = obligation
- $NormActivation = order\_placed(AgY, AgX, GoodsZ)$ denoting that AgY has placed an order to AgX for $GoodsZ$
- $NormCondition = deliver(AgX, AgY, GoodsZ, 1$ week$)$ denoting that AgX has delivers $GoodsZ$ to AgY within 1 week
- $NormExpiration = delivered(AgX, AgY, GoodsZ, 1$ week$)$ denoting that AgX has delivered $GoodsZ$ to AgY in 1 week
- $NormTarget = AgX$

Consider also the permission $\mathbf{Per}_{Del}$ and prohibition $\mathbf{Pro}_{Del}$ that are defined in the same way as $\mathbf{Ob}_{Del}$, except that in the former case $NormType$ = permission, and in the latter case $NormType$ = prohibition. Thus:

- $\mathbf{Per}_{Del}$ permits agent AgX to deliver $GoodsZ$ to agent AgY within one week of receiving the order from AgY
- $\mathbf{Pro}_{Del}$ prohibits agent AgX from delivering $GoodsZ$ to agent AgY within one week of receiving the order from AgY [2]. ■

An administrative architecture [11] has also been defined, consisting of a set of service-oriented middleware components and design patterns to support management of electronic contracts. The architecture can be seen as a combination of the following stages, which are applied to an electronic contracting application as a methodological process. First, off-line verification mechanisms check whether the contracts to be enacted obey certain properties, such as being consistent, or achievable given the possible states the world can reach. The architecture also provides for definition of application specific processes suitable for administration of the electronic contracts through their lifetimes, including enactment, updating, termination, renewal, and so on. Such processes may also include observation of the system, so that the contract can be enforced or otherwise effectively managed. Once suitable application processes are identified, we can specify the roles that agents play within them, and the components that agents can utilise to allow them to manage the contracts. In the following section we give more detail on the agents, components and processes relevant to monitoring.

3 Overview of Monitoring

In this section, we introduce a novel approach to monitoring for on-line detection of contract/norm violations in contract-based systems. The approach describes observation of communications between contract parties, and performs matching of the observed communications against augmented transition networks (*ATN*s)[17] which are essentially directed labelled graphs consisting of nodes and arcs. These *ATN*s characterise acceptable, prohibited, and obliged behaviours. The contract parties are treated as black boxes and their internal state transitions are invisible to the Monitoring components.

[2] The example prohibition is primarily illustrative; it is admittedly somewhat odd to have a prohibition on delivery of goods within a certain time period. We model such a prohibition given the convenient representational match with the permission and obligation.

As discussed in Section 1, our approach to monitoring is based on observation of messages received and sent by agents that are signatories to a contract (the contract parties). This requires that each normative clause in a contract is mapped to a representation whereby:

- the state (of the world) described by a norm's $NormActivation$ can be recognised as being brought about, on the basis of messages exchanged; and
- the state (of the world) described by a norm's $NormCondition$ as being obliged to, permitted to, or prohibited from, being brought about, can be recognised on the basis of messages exchanged.

For example, consider the obligation $\mathbf{Ob}_{Del}$ in Example 1. The obligation is activated when

$$order\_placed(AgY, AgX, GoodsZ)$$

holds, and this is recognised as being the case when the message

$$order(AgY, AgX, GoodsZ)$$

has been sent by AgY to AgX. Observation of this sent message indicates that the contract is in a *critical state* with respect to $\mathbf{Ob}_{Del}$.

The obligation is fulfilled when $\mathbf{Ob}_{Del}$'s

$$NormCondition = deliver(AgX, AgY, GoodsZ, 1 \text{ week})$$

holds. This is recognised as being the case when the message

$$notify\_delivery(AgX, AgY, GoodsZ)$$

is observed as having been sent by AgX to AgY within one week of receipt of the above $order$ message. If the $notify\_delivery$ message is not observed as having been sent within one week, then $\mathbf{Ob}_{Del}$ is deemed to be violated.

In Section 5 we further describe a framework for mapping a contract's normative clauses to *augmented transition networks* (*ATN*s) [17], which constitute the *monitoring representation*, and which associate the $NormActivation$ and $NormCondition$ constituents of norms with messages exchanged.

Figure 1 provides an overview of the monitoring architecture. The architecture has been designed based on the assumption that monitoring parties are external to the contract itself. This means that Monitors can be flexibly deployed for any contracts, provided that appropriate *ATN* representations of contracts are available for input to Monitors, and that Monitors can operate asynchronously from the execution of the contract itself. These design assumptions ensure that the system's performance and monitoring performance are independent of each other. Furthermore, failure of one will not adversely affect the other (for example if one monitor fails then another monitor can be deployed without interrupting execution of the contract).

In the architecture, a *Mapper* maps a contract (obtained from a *Contract Store*) to *ATN*s for input to the *Monitor*. This input is provided off-line. During the run-time enactment of a contract it is the actual messages exchanged that are matched by the Monitor with the *ATN*s in order to detect norm violations.

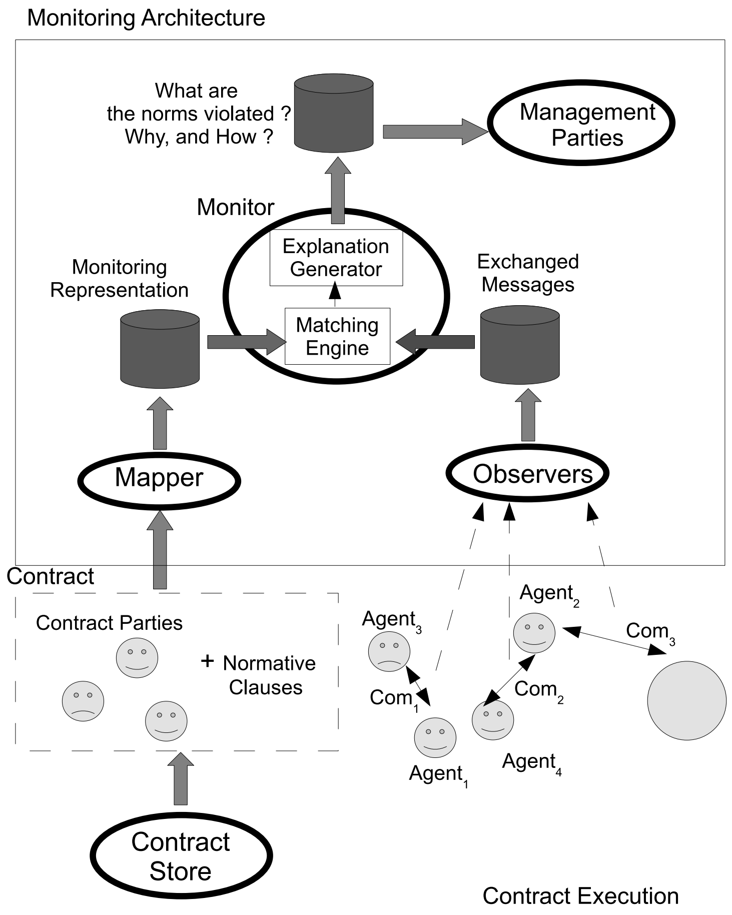


Fig. 1. Monitoring Architecture (ovals denote agents and cylinders denote data stores)

Notice that all messages exchanged between contract party agents, and between contract party agents and the *Environment* (communicative entities that are not contract party agents) must be observable by *Observers*. For effective monitoring, this requirement is mandatory, and can be enforced by enabling Observer interception of all communicative interactions. This is illustrated in Figure 1 in which the Observer *probes* each of the communication channels between agents, and between agents and the environment. These communication channels are conceptual entities; in practice, the probing may be implemented by associating the Observer with the middleware communication interfaces of each agent. Intuitively, if every normative clause is mapped to messages exchanged by entities, and all such messages are observable, then this provides some measure of guarantee that every norm violation can be monitored.

Finally, the Monitor processes each norm violation in order to provide an explanation of the violation that is made use of by *Management* party agents in order to, for example, impose punishments in the case of corrective monitoring, or instigate preemptive action in the case of predictive monitoring. In Section 6 we look forward to future work addressing explanation generation. The following section then characterises the behaviour of the agents in Figure 1 in terms of their interfaces with other agents.

4 Agent Behaviours in the Monitoring Architecture

The monitoring architecture is intended to be integrated into a wide range of applications, and deployed in varying ways. To ease this integration process, we follow a service-oriented approach by defining the form of messages sent from and received by the monitoring components (i.e. their *interfaces*). Publishing an interface allows technology-specific agents to be implemented such that they use the correct format of messages to communicate, regardless of their internal architectures. In this section, we define the interfaces for the components introduced in the previous section. For each component, we define the form of each message type it sends, the parameters the message provides, and the role of the agent expected to receive it. Note that in what follows, messages will contain the unique names *monitor-id*, *observer-id*,. . . of agents and other components of the monitoring architecture.

4.1 Monitor

The Monitor is required to report violations of active contracts (corrective monitoring) and issue warnings when there is a risk of violation (predictive monitoring). These behaviours are specified by messages sent from the Monitor to Observers and Management parties. These messages are of the following form:

– **Subscribe(***monitor-id, observer-id, contract-id, timeInterval/eventExp***)**: The Monitor agent, named *monitor-id*, subscribes to the Observer, named *observer-id* with respect to a given contract, named *contract-id*, in order to receive observed data at intervals corresponding to period, *timeInterval*, or when a condition *eventExp* is true.
– **Cancel(***monitor-id, observer-id, contract-id***)** : The Monitor agent, *monitor-id*, cancels its subscription to the Observer, *observer-id* for *contract-id*.
– **Inform(***monitor-id, manager-id, contract-id, norm, explanation, violator(s)***)** : The Monitor agent, *monitor-id*, informs *manager-id* of a violation of a *norm* by *violator(s)* in *contract-id*, because of *explanation*.
– **Inform(***monitor-id, manager-id, contract-id, norm, explanation, violator(s), timeInterval***)**: The Monitor agent, *monitor-id*, informs *manager-id* of the existence of a danger state in which, after a period of time, *timeInterval*, after the sending of the inform message, *violator(s)* may violate *norm* in *contract-id*, because of *explanation*.

4.2 Observer

The Observer collects data (observes messages). This requires that the Observer subscribes to communication channels between agents, and between agents and the environment.

– **Subscribe**(*observer-id, communication_channel-id*) : Observer, *observer-id*, subscribes to communication channel *communication_channel-id*

Messages are relayed from the Observer to the Monitor:

– **Notify**(*observer-id, monitor-id, contract-id, contract messages*) : Observer, *observer-id*, notifies Monitor, *monitor-id*, of *contract messages* exchanged, sent and received by contract parties in *contract-id*.

4.3 Mapper

The Mapper maps the representation of the contract in a contract store to the *ATN* monitoring representation for input to the Monitor. This behaviour is specified by messages sent from the Mapper to the Contract store and Monitor:

– **Subscribe**(*mapper-id*, *contractStore-id*): *mapper-id* subscribes to *contractStore-id*.
– **Notify**(*mapper-id, monitor-id, contract-id, contractParty-id, transition-structure*): *mapper-id* notifies *monitor-id* with the mapped *ATN* representation, *transition-structure*, of a new active contract, *contract-id*, where this *ATN* specifies *contract messages* (see above) associated with transitions between states, as described in Section 5.

4.4 Contract Store

The Contract Store provides the Mapper with Contracts, as specified by messages sent from the Contract Store to the Mapper. These messages are of the form:

– **Inform**(*contractStore-id, mapper-id, activeContract-id, norm_clauses*): *contractStore-id* provides *mapper-id* with a new contract, *activeContract-id*, and its contained normative content, *norm_clauses*, which is to be mapped to the *ATN*s.

4.5 Manager

The Manager receives the results of monitoring from the Monitor, as specified by messages sent from the Manager to the Monitor:

– **Subscribe**(*manager-id, monitor-id, contract-id*) : *manager-id* subscribes to a Monitor, *monitor-id*, for a contract, *contract-id*.
– **Cancel**(*manager-id, monitor-id, contract-id*) : *manager-id* cancels its subscription to *monitor-id* for *contract-id*.

5 Contract Monitoring: Representation and Interpretation

This section describes the mapping of normative clauses in a contract to its monitoring representation – Augmented Transition Networks (*ATN*s) – such that the normative clauses map to messages exchanged between contract parties. It is these messages that are observed in order to determine when a contract is in a critical state with respect to a given normative clause, and whether contract parties comply with the normative clause.

5.1 Mapping Norms to Augmented Transition Networks

*ATN*s were originally developed for natural language processing, and are recursive in the sense that *ATN*s can themselves label arcs. In the *ATN* representation of normative clauses, nodes correspond to states of the contract in which norms are activated and norms may or may not be violated. Transitions between nodes are labelled by messages sent and received by contract party agents. Intuitively, the messages correspond to actions executed by agents, where these actions in turn bring about states of affairs in which norms are activated, and states of affairs in which norms may or may not be violated.

We have defined a general framework for mapping that takes as input the XML encoding of a contract and its associated OWL (www.w3.org/TR/owl-ref/) encoded domain and action ontologies [15]. Domain ontologies define the predicates used in the description of states, and the action ontologies describe the actions executed by contract party agents and interacting agents in the environment, where these action ontologies include the pre and post-conditions of the actions that are in turn described by predicates in the domain ontology.

Given norm ($NormType$, $NormActivation$, $NormCondition$, $Norm Expiration$, $NormTarget$), then for N = $NormActivation$ or N = $Norm Condition$, the *actors* and *actions* associated with N are identified, and respectively denoted by *actors*(N) and *actions*(N). Currently, this process of identification is not automated, and involves selection of actions in the action ontology whose post-conditions (defined in the domain ontology) match the $NormActivation$ and $NormCondition$.

A mapping is then defined that takes as input N, *actors*(N) and *actions*(N), and returns a set of messages $\mathcal{M}_N$ and synchronisation conditions $Synch_N$ on $\mathcal{M}_N$:

$$message\_map(N, \textit{actors}(N), \textit{actions}(N)) \mapsto (\mathcal{M}_N, Synch_N)$$

Intuitively, messages $\mathcal{M}_N$ are those exchanged between *actors*(N). The synchronisation conditions $Synch_N$ describe temporal relations on these messages such that if the messages are observed as specified by these temporal relations, then one can infer that *actors*(N) have executed *actions*(N) in order to bring about N. Note that in what follows we will focus on the contents of messages and will not commit to a specific agent communication language.

Example 2. Recall $\mathbf{Ob}_{Del}$ in Example 1:

- $NormType$ = obligation
- $NormActivation = order\_placed(AgY, AgX, GoodsZ)$
- $NormCondition = deliver(AgX, AgY, GoodsZ, 1\text{ week})$
- $NormExpiration = delivered(AgX, AgY, GoodsZ, 1\text{ week})$
- $NormTarget = AgY$

For $NormActivation$, the *actors* and *actions* are as follows:

actors($order\_placed(AgY, AgX, GoodsZ)$) = $\{AgY, AgX\}$

actions($order\_placed(AgY, AgX, GoodsZ)$) = $\{order(AgY, AgX, GoodsZ)\}$

and the mapping yields the tuple $(\mathcal{M}_{order\_placed(\ldots)}, Synch_{order\_placed(\ldots)})$ =

$$(\{m1 = order(AgY, AgX, GoodsZ)\}, \{(m1, t1)\})$$

and the synchronisation indicates that $t1$ is the time at which the *order* message $m1$ is sent.

For $NormCondition$, the *actors* and *actions* are as follows:

actors($deliver(AgX, AgY, GoodsZ,$ 1 week)) = $\{AgX, AgY\}$
actions($deliver(AgX, AgY, GoodsZ,$ 1 week)) = $\{deliver(AgX, AgY, GoodsZ)\}$
and the mapping yields the tuple ($\mathcal{M}_{deliver(...)}$,$Synch_{deliver(...)}$) =

$$(\{m2 = notify\_delivery(AgX, AgY, GoodsZ)\}, \{(m2, t1 + 1week\})$$

and the synchronisation indicates that the time at which the $notify\_delivery$ message is sent is within 1 week of the *order* message $m1$ being sent. ■

Notice that both activation of a norm and the norm conditions may involve multiple actors jointly executing actions according to specific temporal constraints. For example, suppose that the obligation on AgX, to deliver $GoodsZ$ to AgY within 1 week, is activated only if AgY has placed the order, *and* within three days of placing the order AgX receives confirmation from its bank that AgY has cleared monies owed to AgX for previous orders. For this activation condition N' we would have:

($\mathcal{M}_{N'}$,$Synch_{N'}$) =
(
$\{m1 = order(AgY, AgX, GoodsZ),$
$m2 = notify\_clearance(Bank\_AgX, AgX, debt(AgY))\},$
$\{(m1, t1), (m2, t1 + 3\text{days})\}$
)

In general, for each normative clause NC in a contract, its $NormActivation$ (NC_A) is mapped to a pair ($\mathcal{M}_{NC_A}$,$Synch_{NC_A}$), which labels a transition to a node (see Figure 2) that denotes a state S which, if the norm is an obligation or prohibition, is critical and so must be monitored. Its $NormCondition$ (NC_C) is mapped to a pair ($\mathcal{M}_{NC_C}$,$Synch_{NC_C}$) that labels a transition from S to S'.

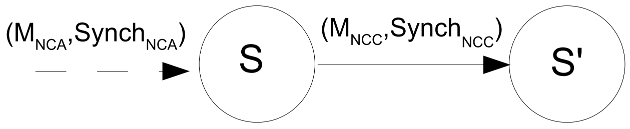


Fig. 2. *ATN* representation of a normative clause

During contract enactment (i.e., at run-time) these ATNs are interpreted with respect to messages observed as defined by the associated synchronisation conditions. In this way, critical states are identified and norm violations detected.

5.2 Interpretation of Augmented Transition Networks

If we refer to Figure 2, then we can make the following general statements:

- If NC is an obligation, then the contract is in a critical state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the obligation is violated if messages $\mathcal{M}_{NC_C}$ are *not* observed as having been sent according to $Synch_{NC_C}$.
- If NC is a prohibition, then the contract is in a critical state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the prohibition is violated if messages $\mathcal{M}_{NC_C}$ *are* observed as having been sent according to $Synch_{NC_C}$.
- If NC is a permission, then the contract is in an allowed state S with respect to NC if messages $\mathcal{M}_{NC_A}$ are observed according to $Synch_{NC_A}$, and the permission is executed if messages $\mathcal{M}_{NC_C}$ *are* observed as having been sent according to $Synch_{NC_C}$. We will further motivate requirements for *ATN* representations of permitted behaviours (where the issue of violation does not arise) in Section 5.3.

Example 3. Recall the obligation $\mathbf{Ob}_{Del}$, permission $\mathbf{Per}_{Del}$ and prohibition $\mathbf{Pro}_{Del}$ described in Example 1. Each of these have the same $NormActivation$ and $Norm$ $Condition$. Hence for each we obtain the *ATN* shown in Figure 3:

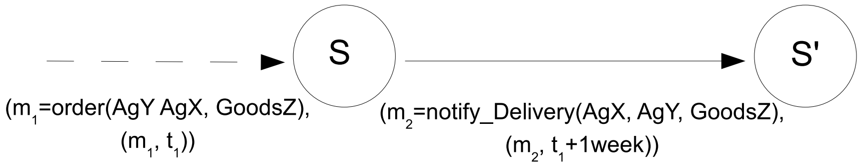

S
S'
(m1=order(AgY AgX, GoodsZ), (m1, t1))
(m2=notify_Delivery(AgX, AgY, GoodsZ), (m2, t1+1week))

Fig. 3. *ATN* representation of $\mathbf{Ob}_{Del}$, $\mathbf{Per}_{Del}$, and $\mathbf{Pro}_{Del}$

In the case that the *ATN* represents $\mathbf{Ob}_{Del}$ or $\mathbf{Pro}_{Del}$, then the contract is in a critical state with respect to the norm if AgX is observed as having received the message $order(AgY, AgX, GoodsZ)$ from AgY at some time $t1$. If the *ATN* represents $\mathbf{Ob}_{Del}$, then $\mathbf{Ob}_{Del}$ is violated if $notify\_delivery(AgX, AgY, GoodsZ)$ *is not* observed as having been sent by AgX to AgY within 1 week after time $t1$. If the *ATN* represents $\mathbf{Pro}_{Del}$, then $\mathbf{Pro}_{Del}$ is violated if $notify\_delivery(AgX, AgY, GoodsZ)\}$ *is* observed as having been sent by AgX to AgY within 1 week after time $t1$. ■

5.3 Composition of *ATN*s

Thus far we have considered only *ATN*s with two nodes representing individual normative clauses. However, a contract may implicitly specify work-flow patterns by associating the $NormCondition$ of one norm with the $NormActivation$ of another. For example, AgX's delivery of $GoodsZ$ to AgY, resulting in the contract state S', may itself be an activation condition for another norm. Hence, if the *ATN* in Figure 3 denotes either $\mathbf{Ob}_{Del}$ or $\mathbf{Per}_{Del}$, then S' may denote a state of the contracts in which the obligation:

AgY is obliged to send payment for $GoodsZ$ to AgX within three days

is activated (illustrating why we want to encode *ATN* representations of permitted behaviours). A transition from S' to S'' will then be labelled by a message sent from AgY to AgX indicating payment. If this message is not observed as having been sent in three days, then the obligation will be deemed violated.

Now, suppose the *ATN* in Figure 3 denotes $\mathbf{Pro}_{Del}$. In this case, observation of the sent message $notify\_delivery(AgX, AgY, GoodsZ)$ within 1 week, indicates violation of the prohibition. S' may then denote a critical state of the contract with respect to a now activated secondary obligation — a *contrary to duty* obligation — which now applies to AgX. Such an obligation might be to pay a penalty that is imposed as a punishment by a management party agent that is informed of AgX's violation of $\mathbf{Pro}_{Del}$.

6 Future Work

We have thus far implemented black box agents and their associated communication interfaces as described in Sections 3 and 4. It remains to further specify and implement the mapping mechanisms outlined in Section 5.1 and implement violation detection algorithms (based on matching *ATN*s and observed messages) and violation explanation algorithms for use by the Monitor.

To enable explanation of violations, we may need to refer to some external representation of the the workflow (that is not implicit in the contract). For example, consider an obligation Ob_1 whereby certain behaviours $Per_1, Per2, \ldots$ are permitted in order to realise this obligation (in planning terms the goal state that is obliged to be realised by Ob_1 may be achieved by plans $Per_1, Per2, \ldots$). Certain behaviours $Pro_1, Pro2, \ldots$ may be prohibited from realising Ob_1. If Ob_1 is detected as having been violated, then an explanation may, for example, indicate that Ob_1 was violated because behaviours $Per_1, Per2, \ldots$ were not executed (as determined by the messages corresponding to these behaviours not being observed), and because behaviours $Pro_1, Pro2, \ldots$ were not allowed for realising Ob_1. Notice that such an explanation could be augmented by domain and situation specific information indicating why $Per_1, Per2, \ldots$ could not be executed. For example consider an obligation to repair an aircraft engine within a given time (this example is taken from the CONTRACT prject use case [10]). In order to fulfill this obligation, it may be permitted to source engine parts from one part manufacturer and prohibited to source engine parts from another part manufacturer. If the obligation is violated (no message notifying completion of repair is sent) then the explanation may account for the fact that the permitted ordering of parts did not take place (augmented by situation specific data as to why the permitted behaviour did not occur).

Finally, we note that the focus of this paper has been on *corrective* monitoring whereby critical states are monitored for violation of norms. *Predictive* monitoring requires representation and recognition of danger states, which are associated with agent behaviours that suggest that a norm may be in danger of violation. Future work will address how such states may be identified empirically, for example by observing and analysing violation of norms at contract run-time and the intermediate states that are reached prior to violation. These intermediate states can then be explicitly included in the *ATN* representation of contracts, so that during future run-time executions, observation of messages indicating transition to these states may signal preemptive action to avoid violation.

7 Conclusions

We conclude with a discussion of closely related work. As mentioned in the introduction, monitoring of contracts has been extensively studied in a Web Services context (e.g., [6,12,13]), where the focus has been on quality of service metrics rather than on the behaviours of contractual entities. Other work has adopted an overhearing approach to monitoring in organisational contexts. Legras et al. [7] use overhearing of messages to monitor changes to the beliefs that agents have about their relationships with other agents in an organisation. Based on this information, a model of how each agent perceives their organisational relationships is accordingly updated. Conversely, Kaminka et al. [5] have developed a plan-recognition approach to overhearing in order to monitor the state of distributed agents that work in a team and collaborate to carry out a specific task. The monitor makes use of the known plan representation of this task to infer on the basis of overheard messages, the belief states of different team-members. These works have thus adopted overhearing in order to infer the mental states of the agents, where these states are domain-dependent and private to the agents. By contrast, in this paper we have described a multi-agent framework that adopts overhearing for a different purpose: it is the states of a contract with respect to its contained norms that are inferred on the basis of messages exchanged. Thus, the proposed approach relies on public knowledge which are norms in a contract. In this way, one can recognise whether a norm is currently in force (activated), in danger of being violated, and whether a norm is in fact violated or complied with. Moreover, in contrast to existing approaches to contract monitoring, our approach benefits from requirements emerging from real world business applications [3].

Acknowledgements. The research described in this paper is partly supported by the European Commission Framework 6 funded project CONTRACT (INFSO-IST-034418). The opinions expressed herein are those of the named authors only and should not be taken as necessarily representative of the opinion of the European Commission or CONTRACT project partners.

References

1. Conte, R., Falcone, R., Sartor, G.: Agents and norms: How to fill the gap? Artificial Intelligence and Law 7, 1–5 (1999)
2. Horling, B., Benyo, B., Lesser, V.: Using self-diagnosis to adapt organizational structures. In: Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, pp. 529–536 (June 2001)
3. Jakob, M., Pchouek, M., Chabera, J., Miles, S., Luck, M., Oren, N., Kollingbaum, M., Holt, C., Vazquez, J., Storms, P., Dehn, M.: Case studies for contract-based systems. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, Industry and Applications Track (2008)

4. Jennings, N.R.: Controlling cooperative problem solving in industrial multi-agent systems using joint intentions. Artificial Intelligence 75(2), 195–240 (1995)
5. Kaminka, G.A., Pynadah, D.V., Tambe, M.: Monitoring teams by overhearing: A multi-agent plan-recognition approach. Journal of Artificial Intelligence Research 17, 83–135 (2002)
6. Keller, A., Ludwig, H.: The WSLA framework: Specifying and monitoring service level agreements for web services. Journal of Network Systems Management 11(1), 57–81 (2003)
7. Legras, F., Tessier, C.: Lotto: group formation by overhearing in large teams. In: AAMAS 2003: Proceedings of the second international joint conference on Autonomous agents and multiagent systems, pp. 425–432. ACM, New York (2003)
8. Lopez, F., Lopez, Y., Luck, M., d'Inverno, M.: A normative framework for agent-based systems. Computational and Mathematical Organization Theory 12(2-3), 227–250 (2006)
9. Mazouzi, H., El Fallah Seghrouchni, A., Haddad, S.: Open protocol design for complex interactions in multi-agent systems. In: AAMAS 2002, pp. 517–526. ACM, New York (2002)
10. Meneguzzi, F.R., Miles, S., Luck, M., Holt, C., Smith, M., Oren, N., Faci, N., Kollingbaum, M., Modgil, S.: Electronic contracting in aircraft aftercare: A case study. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems, Industry and Applications Track (2008)
11. Miles, S., Oren, N., Luck, M., Modgil, S., Faci, N., Holt, C., Vickers, G.: Modelling and administration of contract-based systems. In: Proceedings of the AISB 2008 Symposium on Behaviour Regulation in Multi-agent Systems, pp. 19–24. The Society for the Study of Artificial Intelligence and Simulation of Behaviour (2008)
12. Milosevic, Z., Gibson, S., Linington, P.F., Cole, J., Kulkarni, S.: On design and implementation of a contract monitoring facility. In: Proceedings of the First International Workshop on Electronic Contracting, p. 10. IEEE Computer Society Press, Los Alamitos (2004)
13. Molina-Jimenez, C., Shrivastava, S., Crowcroft, J., Gevros, P.: On the monitoring of contractual service level agreements. In: WEC 2004: Proceedings of the First IEEE International Workshop on Electronic Contracting (WEC 2004), pp. 1–8. IEEE Computer Society Press, Washington (2004)
14. Molina-Jiménez, C., Shrivastava, S.K., Solaiman, E., Warne, J.P.: Contract representation for run-time monitoring and enforcement. In: CEC, pp. 103–110 (2003)
15. Panagiotidi, S., Vazquez-Salceda, J., Alvarez-Napagao, S., Ortega-Martorell, S., Willmott, S., Confalonieri, R., Storms, P.: Intelligent contracting agents language. In: Behaviour Regulation in MAS, AISB 2008 Convention Communication, Interaction and Social Intelligence, pp. 49–55 (2008)
16. Tambe, M.: Towards flexible teamwork. Journal of Artificial Intelligence Research 7, 83–124 (1997)
17. Woods, W.A.: Transition network grammars for natural language analysis. Communications of the ACM 13(10), 591–606 (1970)
18. Xu, L., Jeusfeld, M.A.: Pro-active monitoring of electronic contracts. In: Eder, J., Missikoff, M. (eds.) Advanced Information Systems Engineering (CAiSE), pp. 584–600. Springer, Heidelberg (2003)

Collaborative Load-Balancing in Storage Networks Using Agent Negotiation

Shay Raz[1], Raz Lin[1], and Onn Shehory[2]

[1] Computer Science Department
Bar-Ilan University
Ramat-Gan, 52900 Israel
linraz@cs.biu.ac.il
[2] IBM Research Labs in Israel
c/o Tel Aviv site
Haifa University
Mount Carmel, Haifa 31905 Israel
onn@il.ibm.com

Abstract. The rapid advances in the computer industry allow building larger systems that require mass storage volumes. As storage space increases, its management becomes increasingly difficult. In contemporary enterprise storage systems, performance has developed into a major bottleneck, which negatively affects the overall IT performance. Centralized solutions are often infeasible and thus a distributed solution should be sought. Our novel approach involves incorporating intelligent agents to the storage system, allowing the utilization of a distributed negotiation scheme between agents that act on behalf of the clients who require storage and on behalf of the storage servers within the system. Using a simulation environment which models real settings of a large storage network, we demonstrate the benefits of applying our distributed solution in the storage management domain in terms of client satisfaction, servers' revenue, and overall system performance. Our results show a significant improvement in storage performance when this solution is implemented.

1 Introduction

Advances in contemporary computer and storage technologies allow for networked storage systems to consolidate large storage volumes (in terms of Terabytes) at lower costs[1]. This, in turn, allows storing more data and increasing frequency of data access. At the same time, new data formats (e.g., multimedia formats) impose excessive demands on the I/O rates and storage volumes of storage systems. Whereas these changes are well supported in terms of storage resources, the storage management solutions currently shipped with storage servers are inadequate [12,13,18]. It is becoming evident that a major weakness of contemporary large-scale storage systems is their management. A good solution to storage management should thus be scalable and efficient. In fact, the

[1] E.g., in recent years the storage volume prices have decreased from $1,000,000 per Terabyte in 1992 to $4605.37 in 2001 and to an estimated price of $21.21 in 2010, which translates to a yearly price reduction of 45% [8].

management of storage systems has become a key issue in modern corporate information technology at large.

While in large and complex network systems the use of a centralized approach to system management seems appealing, it is often infeasible. This is due to the large number of diverse devices, and even a larger number of parameters relevant to their management. Consequently, it is rather difficult to ensure a high quality of service for storage clients, without carefully and dynamically adjusting the system to the changing needs. Thus, centralized solutions often attempt to maximize a single objective function, compromising the individual needs of storage consumers and storage providers within the system (e.g., see [21]).

Our solution attempts to address this problem via a distributed model in which intelligent agents cooperate to achieve a shared goal. A distributed storage network can be modeled by a set of clients (applications), requiring storage services, and a set of servers (storage subsystems) which provide these services. A client's I/O operations are typically handled by one of the servers residing in the network, while each server may handle I/O requests from several clients. Note that these I/O requests and storage services can consist of both *read* and *write* operations.

Thus, in this paper we present a storage management mechanism. Our mechanism overcomes limitations of current approaches. In our solution, a distributed negotiation mechanism serves as a means for managing storage networks. We investigate the benefits of implementing such negotiations between servers and clients in terms of client satisfaction, servers' revenue, and overall system performance. By adding negotiation capabilities to storage consumers and providers, we enable dynamic load-balancing of the servers' load, thus increasing both the revenues for the servers, and the satisfaction of the clients. We note, however, that the clients and the servers have different maximization objectives, which may be contradicting. While the servers aim to maximize their revenues from the clients, the clients aim to maximize their satisfaction from the service.

We evaluate our solution via experiments. Using a simulation environment which models a realistic setting of a storage network, we demonstrate the efficacy of the distributed negotiation scheme in these types of storage systems. In addition, we demonstrate the generic nature of this protocol by experimenting with different types of actions taken by both the servers and the clients.

In a nut shell, this paper advances the state-of-the-art in several ways. It demonstrates the efficacy of distributed negotiations in online load-balancing of large storage networks. Traditional load-balancing techniques for storage systems are commonly centralized, and even when distributed, do not address the case in which the storage subsystems are autonomous and possibly represent self-interested parties [7]. Our approach, on the other hand, allows, via distributed negotiation, to aspire towards better load-balancing, while preserving the autonomy of each subsystem. In addition, given the importance of managing large storage systems, performance becomes a key factor for effective storage. Our novel approach allows the improvement of performance of storage networks in a timely manner and allows online adaptation to changing demands, which are characteristics of computer systems. Lastly, in contrast to existing solutions [7], we recognize two distinct types of players in the storage domain: the consumers (clients) and the storage providers (servers). Our model takes this into account since each type of player has a different objective and is modeled differently.

The remainder of the paper is organized as follows. Section 2 provides an overview of the problem and the storage domain. Section 3 surveys related work in the field of distributed negotiation and storage performance. Section 4 describes our negotiation protocol and the evaluation's mechanism of both the server and the client. Section 5 describes the simulation design while Section 6 describes the experimental setting and methodology and reviews the results. Finally, we conclude in Section 7.

2 Problem Context: Storage Networks

Consider a distributed storage network comprised of multiple storage clients and servers. Each client generates I/O requests and expects a certain quality of service, and storage servers service these requests, for some monetary compensation. When servers fail to address requests to which they have committed at the agreed upon quality, they may be subject to penalties. In our model, at a specific time period, each client sends its storage requests to a single storage server, yet the client is also free to switch to another server to better address its needs at proceeding time periods. Servers may handle several requests concurrently. This interaction and compensation model among the clients and the servers is formally expressed by service level agreements (SLA) and service level objectives (SLO) ([16], Part I). The SLA states that partial fulfillment of the quality of service (QoS) implies partial payment by the client, while the SLO is the server's service objective to ensure the minimum QoS.

Formally, we have a set of clients $Cl = \{c_1, c_2, \ldots, c_n\}$ and a set of servers $Sr = \{s_1, s_2, \ldots, s_m\}$. Let $REQS$ be the set of all possible I/O requests. Note that $\perp \in REQS$. Let $c_j^{reqs}(it) \subseteq REQS$ denote the set of I/O requests generated by client $c_j \in Cl$ at iteration it. A single iteration consists of several fixed time segments in which negotiation-related actions can be made.

The clients are modeled using an open subsystem model (e.g., see [6]). That is, new requests are generated independently of the completion time of the previous requests. Clients differ from one another in the intensity of their I/O requests. A client with an intensity value of 0 basically does not require any I/O services, while the higher the intensity value of a client the more I/O requests it generates in a given time unit. Thus, the I/O intensity of a client corresponds to its typical pattern of I/O usage. For instance, a client that uses video streaming has a large I/O intensity, whereas a client that performs text editing has a low I/O intensity. The client's intensity reflects the time density of its service requests (e.g., a client with an intensity value of 2 will require twice as much I/O requests in the same time segment as compared to a client with an intensity value of 1). We denote the intensity of a client c_j as c_j^{int}. In our experiments, we inflict several I/O intensities on the clients involved.

In our model, clients pay for the services using tokens. Each client is allocated, at each iteration, a fixed number of tokens to be used during that iteration. We denote the number of tokens of client c_j as c_j^{tok}. Each client c_j maintains a satisfaction level, denoted $c_j^{sat}(it)$, which reflects its satisfaction level from the quality of service it has received at iteration it, such that $0 \leq c_j^{sat}(it) \leq 1$. This value is calculated using a linear combination of the average request time and its standard deviation. The actual payment made by the client at iteration it is denoted by $c_j^{pay}(it)$. Note that the payment is made by the client per service it has received from the server within a fixed period of time, and it is computed in correspondence to the client's satisfaction level from that service. Thus, the actual payment, $c_j^{pay}(it)$, equals $c_j^{tok} \cdot c_j^{sat}(it)$, such that $0 \leq c_j^{pay}(it) \leq c_j^{tok}$.

In our model, all servers are assumed to be of an equal type. That is, each has the same physical characteristics and capabilities. Note that this does not prevent them from having different load preferences, as presented later in our experiments. The server's address space is divided into K equal segments, each being a logical partition. Thus, a single server can serve up to K clients, by allocating a partition space to each. The service provided by the server is for I/O operations. The server's load is measured in terms of queue length, which is a well known estimator of the load on a storage device (e.g., [20]). The server's average queue at time period t is denoted $s_i^{queue}(t)$, and integrated over that time period. Formally, if t_{it} denotes the start time of iteration it, then s_i's average queue length in it, denoted $s_i^{AvgQueue}(it)$, is:

$$s_i^{AvgQueue}(it) = \frac{\int_{t_{it}}^{t_{it+1}} s_i^{queue}(t)dt}{t_{it+1} - t_{it}} \qquad (1)$$

The objective of the storage management system is to maximize the system's utility, while keeping the performance level sufficiently high to meet the clients' requirements. To prevent deterioration in the quality of service, the system should be equipped with means for efficient matching between servers and clients, such that the servers are able to complete the requests of the clients within a sufficiently short time to meet their SLAs with their clients. When meeting their SLAs, the servers will receive full payment from their clients. In such cases, the client will pay all of its tokens, since its satisfaction is 1. That is, in such cases client c_j will pay $c_j^{pay}(it) = c_j^{tok}$.

Systems of the sort presented above, in particular ones that consist of numerous clients and servers, can benefit from a distributed solution. Such a solution has the advantages of avoiding a single point of failure, and allowing the storage management mechanism to be functional even if some nodes malfunction. Hence, our goal is to provide a distributed mechanism which ensures acceptable storage performance: once a performance problem is identified, our mechanism should dynamically and proactively resolve it. The mechanism should allow clients to efficiently locate servers suitable for their needs, and servers to offer and publish their capabilities to potential clients.

We proceed by reviewing current storage management solutions, as well as relevant studies in the field of distributed negotiation.

3 Related Work

In recent years, research has demonstrated the promise of using cooperative negotiation approaches in real-time load balancing (e.g., in cellular domains see [1,10]; in electricity usage see [2]). In the context of storage system networks, a negotiation model is also presented in [5,15]. Stoupa and Vakali [15] present a QoS negotiation model which is client-oriented. The model itself allows the clients to describe their characteristics and rank the importance of the different QoS parameters. Then, the system matches them with a list of potential storage subsystems. The negotiation terminates when the client chooses a specific subsystem. In contrast to [15], we propose a distributed mechanism, and not a centralized one. A distributed approach should turn out to be more effective in overcoming bottlenecks and avoiding a single point of failure. Our algorithm also inherently allows changing the assignments of clients to servers based on servers' load. Moreover, a distributed mechanism is more appropriate for large networks, in which maintaining a centralized management agent will be more difficult and resource consuming. In addition, in our model both the clients and the servers take part in the negotiation, while in their model only the clients take an active part.

Czajkowski et al. [5] describe an agent based negotiation system for computer resource allocation in a distributed network. Similar to our work, the negotiation is done to ensure high levels of SLAs. However, they introduce centralized coordination management functions which are used to coordinate the different tasks. We, on the other hand, adopt a distributed approach, which should be more effective and more scalable in large systems. Furthermore, their method has not yet been validated, while we show the efficacy of our work using simulations.

Xu et al. [20] address the case of load-balancing of a distributed storage system. Their objective, though, is to achieve better performance of the system, where the system is modeled as a whole. We, on the other hand, address the case of a system comprised of multiple clients and servers, where each has its own utility function to be maximized. We do, however, maximize the system utility in cases where this utility is additive. We also assert that the objectives of the clients and the severs may be conflicting. Thus, we try to achieve a load-balance in such a way that will take this into consideration, while still achieving better performance of the system as a whole.

In the context of modeling the storage devices, the usage of both numerical simulators and analytical models are common. Disk simulators, such as Pantheon [19] and DiskSim [3], which are effective in simulating storage device behavior, are being used. These simulators produce accurate per-request response times. Analytical storage device models (e.g., see [4,9,11,14,17]) are somewhat simpler to compute since they describe device behavior via a set of formulas. However, finding a reliable formula set can be a very difficult task, as it requires comprehensive knowledge and understanding of the interaction between storage devices and workloads. We have therefore elected to use a simulator, and specifically, DiskSim [3]. The use of DiskSim has enabled us to simulate a real disk settings[2], and to incorporate our novel distributed approach into the system level layer.

4 Model and Protocol

We first present the formal model of the problem addressed, and then the distributed negotiation protocol.

[2] DiskSim simulates many components of the disk system, such as: disks, controllers, buses and cache.

4.1 Formal Model

We consider a distributed negotiation protocol between storage clients and servers. We assume that the clients' SLAs and the servers' SLOs are common knowledge. The sought agreement between the clients and the servers should match the former to the latter.

In our model, participants negotiate to maximize three types of utility functions: for clients, for servers, and for the whole system. The utility function of a client measures its satisfaction level based on the service it receives. We express it by a weighted average of the average request service time and its standard deviation.

There are two types of request service times in our utility model: the service time sought and expected by a client, as expressed in its SLA, and the actual service time, as measured in experiments. We denote for each iteration it the average SLA service time by $T_{Avg}^{SLA}(c_j^{reqs}(it))$, and the average actual service time by $T_{Avg}^{actual}(c_j^{reqs}(it))$. Similar terms are used for the corresponding average standard deviations (replace $_{Avg}$ by $_{AvgStd}$).

Formally, the client's utility function is given by the following formula:

$$u_{c_j}(it) = A \cdot \min\{1, \frac{T_{Avg}^{SLA}(c_j^{reqs}(it))}{T_{Avg}^{actual}(c_j^{reqs}(it))}\} + B \cdot \min\{1, \frac{T_{AvgStd}^{SLA}(c_j^{reqs}(it))}{T_{AvgStd}^{actual}(c_j^{reqs}(it))}\} \tag{2}$$

such that $A + B = 1$. In the simulations we set A and B to 0.5.

The utility function of the server measured its revenues from providing the services. We denote the set of clients serviced by server s_i during iteration it as $Cl_{s_i}(it)$, such that $Cl_{s_i}(it) \subseteq Cl$. The utility of s_i is given by:

$$u_{s_i}(it) = \sum_{c_j \in Cl_{s_i}(it)} c_j^{pay}(it) \tag{3}$$

The system-level utility, which is calculated per iteration and denoted u_{sys}, considers all of the requests made in the system during a given iteration. It is a weighted average of the clients' utilities, as follows:

$$u_{sys}(it) = \frac{\sum_{j=1}^{|Cl|}(c_j^{tok} \cdot u_{c_j}(it))}{\sum_{j=1}^{|Cl|} c_j^{tok}} \tag{4}$$

An agreement $\boldsymbol{a} \in A$ is a mapping between a client and a server, which conforms to the SLA between them, that is, $\boldsymbol{a} = \{c_j, s_i\}$, where $c_j \in Cl, s_i \in Sr$. Since a client can choose not to work with any server and a server can choose to remove a client from its partition, we also have a special agreement $\{c_j, \perp\} \in A$.

4.2 The Distributed Negotiation Protocol

The negotiation itself can be triggered by different events (e.g., global events as time interrupt or local events, such as exceeding load thresholds). The specific trigger to be used is principally external to our mechanism. In our simulations we used predefined time-unit intervals. In each iteration of our simulations servers and clients are selected randomly to engage in the negotiations.

At first, the clients are not associated with any server and can choose to (a) request a service from a server, or (b) disengage from its current server. The server can choose to (a) offer its services to a specific client, or (b) remove a client from its partition. Below we elaborate on these different actions.

Server Actions. The server's actions depend on its estimated average queue length at iteration $it + 1$, denoted ${s'}_i^{AvgQueue}(it + 1)$. This estimation takes into account the server's current average queue length, as well as the clients the server handles. As some client might have been removed from the server's partitions during the iteration, while others might have been added, the server has to estimate the intensity of the clients it will handle. To this end, the server uses a linear estimation of its future load according to its current load. Due to lack of space we will not describe these formulas in detail.

The server has two thresholds which determine its load status - an upper bound threshold and a lower bound threshold. If the server's estimated average queue length is above the upper bound threshold then the server is over-loaded. In this case, the server chooses to remove a client c_j which is allocated to one of its partitions. The agreement $\{c_j, \perp\}$ is automatically implemented. If the server's estimated average queue length is below the lower bound, then it is under-loaded. In this case, the server tries to find a client c_j that has not been allocated to any partition in the system. If such a client is found, the server initiates a request for a commitment offer from the client. If the client accepts the request, then an agreement $\boldsymbol{a} = \{c_j, s_i\}$ which adheres to the SLA between c_j and s_i is reached. While the server's decision making process regarding which client to select is principally external to our proposed mechanism, in our simulations we modeled only one behavior type in which the server selects a vacant client arbitrarily.

Finally, the server can be in a stable load, if its estimated average queue length is in-between. When in stable load, and in order to avoid local minima and with the prospect of increasing the server's utility in the next iteration, the server may choose, with a probability of Pr_{rem}, to remove a client c_j from its partition and to implement the $\{c_j, \perp\}$ agreement. Pr_{rem} is set such that it ensures that, at most, half of the time, the server will choose the removal action. Formally:

$$Pr_{rem} = \min(0.5, \frac{vacant}{|Sr|}) \tag{5}$$

where $vacant$ represents the number of clients that are not mapped to any server. The probability will converge to 0.5 in cases in which there are many vacant clients. This will allow the servers to try to find more profitable clients at the expense of other clients. Nonetheless, we limit the probability to a maximum value of 0.5 in order to control the stabilization of the system.

In our simulations, the least profitable client for the server was removed. The profitability of a client is measured as the ratio between client c_j's potential maximum payment and its generated load during the current iteration it, that is $\frac{c_j^{tok}}{c_j^{req}(it)}$.

Note that our utility maximization is based on a heuristic approach, which entails that servers drop a client when over-loaded or with a stable load. While this heuristic can cause a temporary decrease in the server's utility, it can also lead to an increase in the utility in the long run. Our experiments indeed show (Section 6) the efficacy of this heuristic.

Client Actions. The client's actions depend on whether it is already mapped to a server or not. If a client c_j is not mapped to a server, then it tries to find a server from which to request services, based on its SLA. In the simulations we modeled only one behavior in which the client selects an arbitrary server. If a server s_i is found, then the client initiates a request for commitment to the offer of that server. If the server accepts the request, then an agreement $\{c_j, s_i\}$ is reached and the mapping is implemented.

If the client is already receiving services from a server, the client calculates its satisfaction from the service. Then, the client disengages from its server using the following probability measure, based on a given satisfaction threshold thr_{sat}:

$$Pr = MRF \cdot \max\{0, \frac{thr_{sat} - c_j^{sat}(it)}{thr_{sat}}\}, 0 \leq MRF \leq 1 \tag{6}$$

where MRF represents the client's tolerance to an inadequate quality of service. Higher values of MRF will increase the probability that an unsatisfied client will remove itself from the server, while lower values decrease this probability. A client that decides to remove itself from the server implements the $\{c_j, \perp\}$ agreement.

5 Simulation Design

The objective of our simulations was to show that our novel distributed approach enables an efficient load-balancing of the storage system. A building block of the simulation system was the DiskSim simulator [3], which is an efficient and highly configurable disk system simulator. Additional designs and implementations were necessary to build a multi-disk distributed storage system. The simulation was divided into fixed time segment iterations. Each iteration consisted of a simulation phase followed by a negotiation phase. The service level parameters, such as request time and load parameters, were calculated for each iteration.

While our model permits any kind of servers and clients, in the simulations we had to limit this diversity for practical reasons. Specifically, we allowed four different types of servers and four different types of clients. These types represent typical and common characteristics of servers and clients. Recall that the server's load is measured in terms of queue length. Thus, each type of server can satisfy a different quantity of I/O requests per time unit. We model this with a lower and upper bound thresholds, which state the server's load status. With regard to the clients, each type of client had a different type of SLA requirement. Each client differed by its intensity value. Table 1 describes the different types of servers and clients that were used. Correlations existed between client and server types, such that each client type would be best served by a specific server type. As clients differ by their I/O intensity values and the servers by their thresholds for under-load states and overload states, the best correlation is achieved when the I/O

Table 1. Servers' types measured by the average queue length and matched clients' types and intensities

Server Type	Lower Bound (under-load)	Upper Bound (overload)	Client Type	Client's Intensity
Alpha	0.1	0.2	*A*	0.3
Beta	0.2	0.5	*B*	0.45
Gamma	0.5	1.5	*C*	0.55
Delta	1.5	15.0	*D*	0.65

intensities of the clients allow the servers to be in a stable load, that is in-between their load boundaries. For example, a client of type A would be best served by the Alpha server, as the load imposed by it and the quality of service provided by the server correspond.

We ran five sets of experiments in order to test our mechanism. In each experiment 6 servers and 30 clients were used. In each experiment we measured the effects of the MRF parameter on the success of load-balancing the system using our model. Recall that the MRF parameter is an important parameter as it influences the client's tolerance toward an inadequate quality of service. Thus, the MRF parameter can have a substantial effect on the stability of the system. We hypothesized that higher MRF values would lead to lower system utility values, and our results indeed support this hypothesis. Thus, after checking the whole MRF range, we proceeded and ran several runs per experiment, focusing on lower MRF values — those proven to be better in our earlier experiments. Each experiment consisted of 10 runs using different random seeds of 50 iterations each.

Each experiment was performed with a different setting of clients and servers. In the first experiment we tested the benefits of our algorithm when only clients take an active part in the negotiations. Thus, 6 servers with no lower bound threshold were simulated. Note that we ran all of the experiments with different MRF values. When the MRF value is 0, it means that the client will not be active in the negotiations (i.e., it will not try to remove itself from the server to which it is attached), regardless of its satisfaction level from that server. Thus, the results of the first experiment and the runs in which the MRF value is equal to 0, are important as they demonstrate the efficacy of our negotiation model in cases where only one side (clients or servers) perform negotiations. This is an important property since it is not always possible to control the behavior of all parties in all distributed systems.

The second experiment consisted of three types of servers and clients, while the third experiment consisted of four types of each. These different settings were chosen to demonstrate the efficacy of our model when there is a variety in the characteristics of the servers and the clients. In this manner we were also able to show the competitiveness of our model compared to the optimal theoretical solution. In these specific settings, the optimal solution would yield a utility of 1.

The fourth experiment consisted of four types of servers and clients as well (as in experiments 3), yet the number of servers of each type was chosen randomly. Lastly, we compared our distributed algorithm with a greedy algorithm. In the greedy algorithm, any client can choose to engage with any available server, and any unsatisfied client can disengage from its server and engage with a less loaded server which has a vacant partition. Table 2 describes the different numbers of each client and server types in the different experiments[3].

Table 2. Simulation settings

	Client's Type				**Server's Type**				
#	*A*	*B*	*C*	*D*	*Alpha*	*Beta*	*Gamma*	*Delta*	unbounded
1	5	10	10	5	0	0	0	0	6
2	10	10	10	0	2	2	2	0	0
3	5	10	10	5	1	2	2	1	0
4	5	10	10	5	1	1	2	2	0
5	0	10	10	10	0	2	2	2	0

6 Simulation Results

Figure 1 depicts the change of the system's utility value for different values of MRF, as a function of the iteration number, for the first experiment, in which the servers are passive in the negotiations. The results show that our mechanism appears to be useful and efficient even in cases where only the clients take an active part.

As illustrated in Figure 1, when the servers do not implement any load-balancing strategy and the clients apply our strategies results in the sought improvement in the system's utility. We can also observe that in the absence of our mechanism, the average system's utility is 0.55. Note that this is a generic result which does not refer to a specific system configuration (as it is an average of the multiple random configurations). Using our mechanism we can see that the client's load balancing improves significantly to more than 0.75. Of course, for a specific system, one could devise a specific configuration that would improve the 0.55 utility value, and possibly the 0.75 value as well. However, this would require a careful planning, and most likely would also involve manual design and fine-tuning, while an automated design could actually be performed. Indeed, our mechanism can serve as a design tool for such specific configurations.

As we mentioned earlier, additional three scenarios were used to test our algorithm. Figures 2, 3 and 4 demonstrate the change of the system's utility value for different values of MRF, as a function of the iteration number, for the three types, four types and four random types of servers and clients, respectively.

We can see that in all three scenarios, our algorithm quickly achieves higher utility values (e.g., when the MRF value is set to 0.1 the average utility value is 0.84, 0.82 and 0.80 for the three types, four types and four random types of servers and clients, respectively) than the average system's utility of 0.55 when our mechanism is not implemented. We can also see that in most cases, smaller MRFs values indeed yield better

[3] We also ran simulations with two types of servers and clients and achieved good results. However, since in real settings there are rarely only two types of each, we do not present these results in this paper.

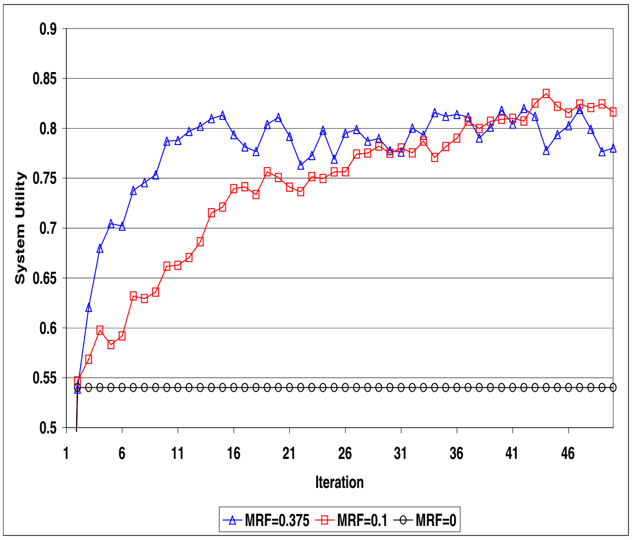


Fig. 1. System utility in servers with no lower-bound threshold

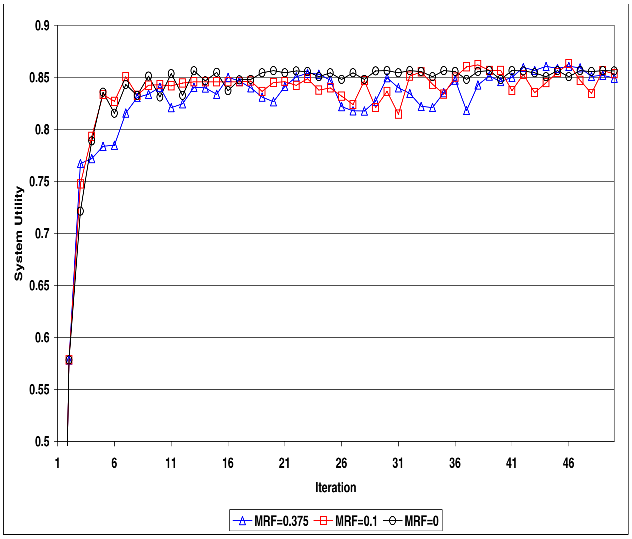


Fig. 2. System utility for the three types of servers and clients

overall results. Even in the special case in which the MRF value is set to 0, that is, the clients are passive in the negotiations, the system's utilities are much higher than 0.55.

It is interesting to note, though, the special behavior achieved when the MRF value is set to 0, as demonstrated in Figure 4. In this case, a client receiving services, will not choose the action to be removed from the server to which it is connected. This is regardless of the client's satisfaction level from that server. Thus, the system's utility is more likely to converge, as clients tend to remain connected to their servers, despite the QoS they obtain, as indeed shown in Figure 4 (a system utility value of 0.75). However, this can lead to a local maximum and prevent the system from achieving higher utility values (as indeed demonstrated in the case of $MRF = 0.1$).

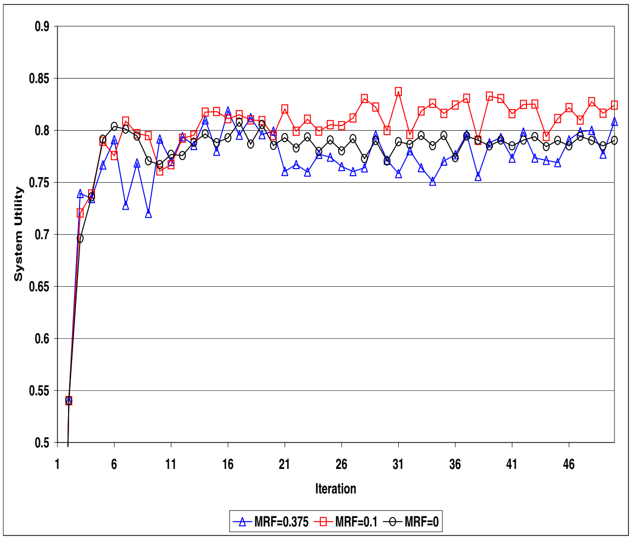


Fig. 3. System utility for the four types of servers and clients

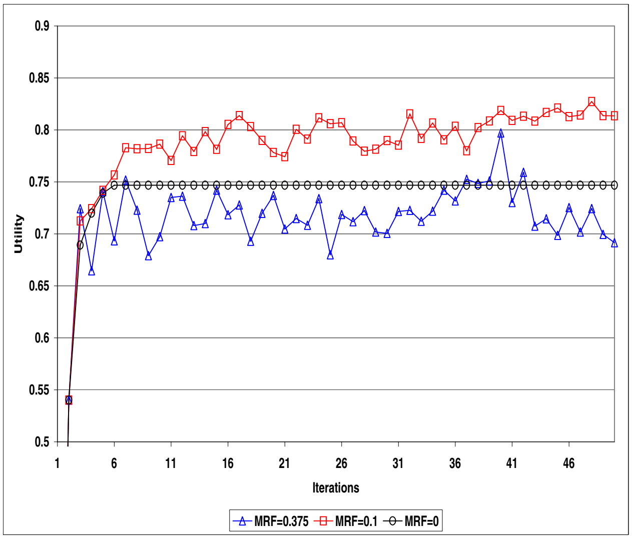


Fig. 4. System utility for four random types of servers and clients

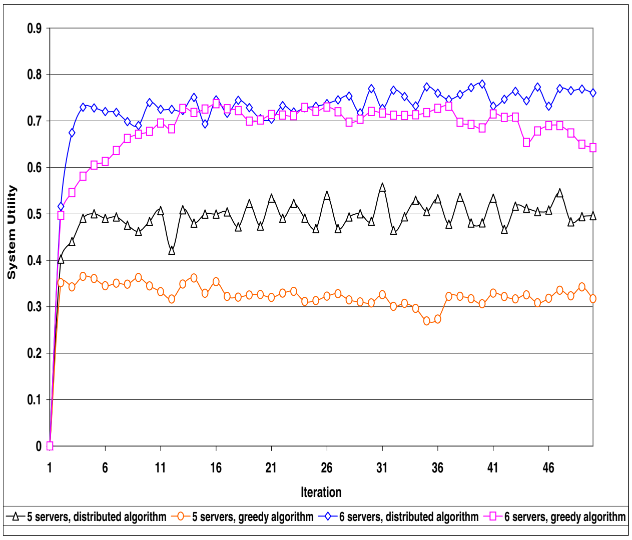


Fig. 5. System utility for the greedy and distributed algorithms

Figure 5 presents the results of the greedy algorithm and our distributed algorithm in two distinct scenarios. In both we have three clients of type B, C and D. Yet, in the first scenario we have 6 servers, while in the second we have 5 servers which represent a scenario in which there is a shortage of resources (that is, heavier load exists). In the first scenario our algorithm performs as good as the greedy algorithms, yet the greedy algorithm converges more slowly than our algorithm. In the second scenario our algorithm generates better results. While our results show that there is not much benefit to our mechanism over the greedy one in cases in which the load of the system is negligible (as reflected by the 6 servers scenario), our algorithm outperforms the greedy algorithm in those cases it is most needed, that is, when heavy load exists. These are the cases when good load management is crucial for continuous usage of the storage system.

Based on these experiments we can see that our mechanism is beneficial when multiple agents (clients and servers) of different types are involved, as well as in cases in which only one side takes an active part in negotiations. Our mechanism allows the system to achieve better performance in real-time while avoiding local maxima and maximizing the satisfaction of the clients.

7 Conclusions

This paper presents a novel approach which allows storage systems to efficiently manage their resources using distributed negotiations. Our mechanism does not impose heavy computational or network overheads on any single unit within the system. Therefore, it should scale well. Additionally, our mechanism allows each participant to seek maximization of its own utility. Thus, we overcome the limitation presented by the common approach to distributed storage resource allocation, where only the overall system utility is considered.

Using a simulation environment, we have shown how dynamical load-balancing of the servers' load enables an increase in both the servers' revenues and the clients' satisfaction. We have shown that our mechanism, due to its distributed nature, has proven to be useful and efficient even in cases where only clients, or only servers, implement it. Moreover, we have shown that it can outperform the greedy algorithm in situations in which solutions are more difficult to arrive at. In addition, our simulation system can be used as a design tool for a system administrator to help provide a better load-balancing scheme prior to first system initialization.

Motivated by the promising results obtained thus far, future research should focus on implementing different behaviors for the servers as well as the clients. These behaviors may help in achieving even better performance overall. Furthermore, it may be helpful to allow servers to dynamically adapt their thresholds, based on the given load in the entire system.

References

1. Bigham, J., Du, L.: Cooperative negotiation in a multi-agent system for real-time load balancing. In: Proceedings of AAMAS, pp. 568–575 (2003)
2. Brazier, F., Cornelissen, F., Gustavsson, R., Jonker, C.M., Lindeberg, O., Polak, B., Treur, J.: Agents negotiating for load balancing of electricity use. In: Proceedings of 18th International Conference on Distributed Computing Systems, pp. 622–629 (1998)
3. Bucy, J., Ganger, G.: The DiskSim simulation environment version 3.0 reference manual. Technical Report CMU-CS-03-102, Carnegie Mellon University (2003)
4. Chen, S., Towsley, D.: A performance evaluation of raid architectures. IEEE Transactions on Computers 45(10), 1116–1130 (1996)
5. Czajkowski, K., Foster, I., Kesselman, C., Sander, V., Tuecke, S.: Snap: A protocol for negotiating service level agreements and coordinating resources management in distributed systems. In: Proceedings of the 8th Workshop on Job Scheduling Strategies for Parallel Processing, pp. 153–183 (2002)
6. Ganger, G.R.: System-Oriented Evaluation of I/O Subsystem Performance. PhD thesis, University of Michigan (1995)
7. Ganger, G.R., Wothington, B.L., Hou, R.Y., Patt, Y.N.: Disk arrays: High-performance, high-reliability storage subsystems. IEEE Computer 27(3), 30–36 (1994)
8. Gilheany, S.: Projecting the cost of magnetic disk storage over the next 10 years. White paper, Berghell Associates (January 2001)
9. Lee, E.K., Katz, R.H.: An analytic performance model of disk arrays. In: Proceedings of ACM SIGMETRICS, pp. 98–109 (1993)
10. Lin, R., Dor-Shifer, D., Rosenberg, S., Kraus, S., Sarne, D.: Towards the fourth generation of cellular networks: Improving performance using distributed negotiation. In: Proceedings of the 9th ACM international Symposium on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pp. 347–356 (2006)
11. Merchant, A., Alvarez, G.A.: Disk array models in Minerva. Technical Report HPL-2001-118, HP Laboratories (2001)
12. Morris, R.J.T., Truskowski, B.J.: The evolution of storage systems. Storage Systems 42(2), 205–217 (2003)

13. Santos, J.R., Mutz, R.: Performance analysis of the rio multimedia storage system with heterogeneous disk configurations. In: Proceedings of 6th ACM International Conference on Multimedia, pp. 303–305 (1998)
14. Shriver, E., Merchant, A., Wilkes, J.: An analytic behavior model for disk drives with readahead caches and request reordering. In: Proceedings of International Conference on Measurement and Modeling of Computer Systems, pp. 182–191 (1998)
15. Stoupa, K., Vakali, A.: QoS-oriented negotiation in disk subsystems. Data & Knowledge Engineering 58(2), 107–128 (2006)
16. Sturm, R., Morris, W., Jander, M.: Foundations of Service Level Management, 1st edn. Sams, Indianapolis, Ind (2000)
17. Uysal, M., Alvarez, G.A., Merchant, A.: A modular, analytical throughput model for modern disk arrays. In: Proceedings of 9th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems, pp. 183–192 (2001)
18. Vogel, A., Kerhervé, B., von Bochmann, G., Gecsei, J.: Distributed multimedia applications and quality of service - a survey. IEEE Multimedia 2(2), 10–19 (1995)
19. Wilkes, J.: The Pantheon storage-system simulator. Technical Report HPL-SSP-95-14, HP Laboratories (1995)
20. Xu, Z., Zhu, Y., Min, R., Hu, Y.: Achieving better load balance in distributed storage system. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 1314–1322 (2002)
21. Yin, L., Uttamchandani, S., Korupolu, M., Voruganti, K., Katz, R.: Smart: An integrated multi-action ddvisor for storage systems. In: Proceedings of the USENIX Annual Conference, pp. 229–242 (2006)

Author Index